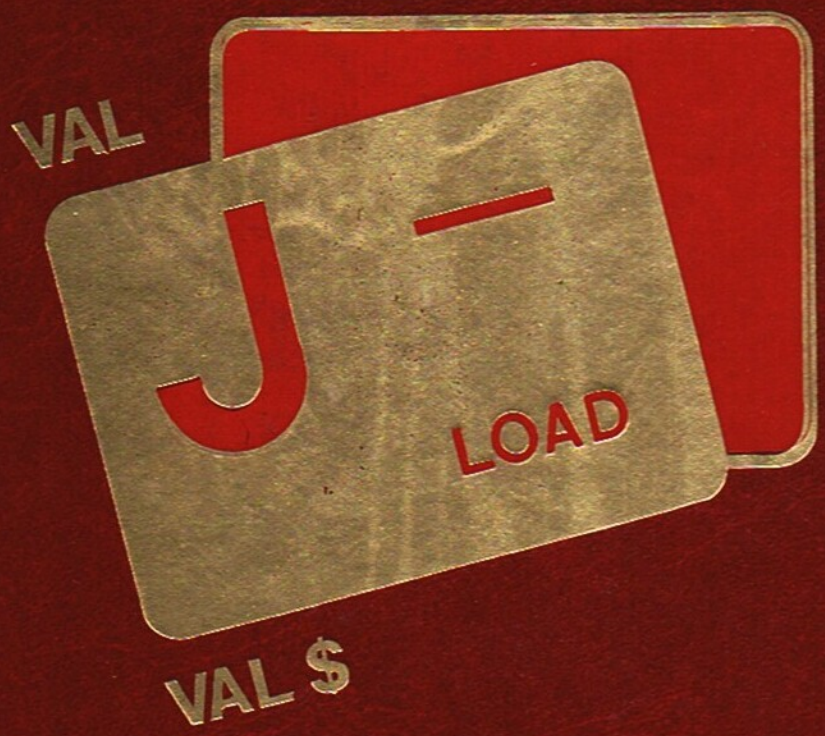
RUN

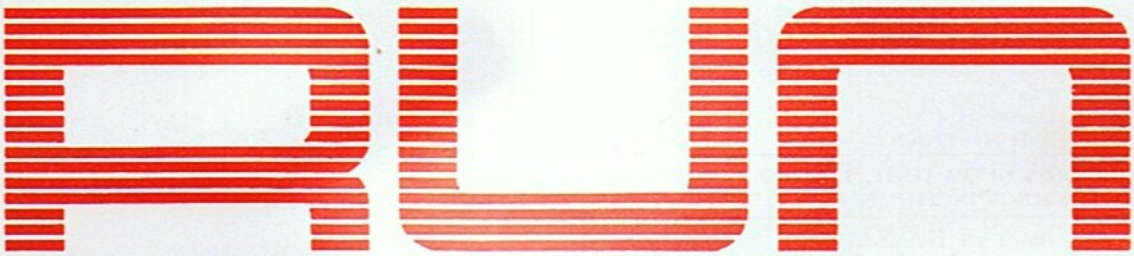
Enciclopedia Práctica del Spectrum



Nueva Lente/Ingelek

RUN

Enciclopedia Práctica del Spectrum



Enciclopedia Práctica del Spectrum

Nueva Lente/Ingelek

Una publicación

EDICIONES NUEVA LENTE, S. A.
y EDICIONES INGELEK, S. A.

Director-editor por Nueva Lente, S. A.:
MIGUEL J. GONI
Director-editor por Ingelek, S. A.:
ANTONIO M. FERRER
Director de producción:
RICARDO ESPAÑOL
Jefe de producción:
SANTOS ROBLES

Director de la obra:
FERNANDO LOPEZ MARTINEZ
Coordinación:
VICENTE ROBLES
Colaboradores:
JUAN MANUEL LOPEZ MARTINEZ
CARLOS DE LA OSSA
JUAN MANUEL MAYORAL

Diseño Gráfico:
JOSE OCHOA
Maquetación:
CARLOS GONZALEZ-AMEZUA
Fotografía:
«Equipo Gálata»
(ALBINO LOPEZ y EDUARDO AGUDELO)
Ilustraciones:
JOSE OCHOA
ALFONSO MENDEZ
ANTONIO PERERA

© Ediciones Nueva Lente, S. A.
Madrid 1985
Dirección, redacción y administración:
Benito Castro, 12. 28028-Madrid
Teléfs.: 245 45 98 y 246 73 67

© Ediciones Ingelek, S. A.
Madrid 1985
Dirección, redacción y administración:
Avda. Alfonso XIII, 141. 28016-Madrid
Teléf.: 250 58 20

ISBN tomo primero: 84-7534-120-9
ISBN tomo segundo: 84-7534-121-7
ISBN tomo tercero: 84-7534-122-5
ISBN tomo cuarto: 84-7534-123-3
ISBN de la obra: 84-7534-118-7

Fotomecánica: OCHOA, S. A.
Ricardo Ortiz, 74. 28017-Madrid

Impresión: GRAFICAS REUNIDAS, S. A.
Avda. de Aragón, 56. 28027-Madrid

Depósito legal: M-9896-1985
PRINTED IN SPAIN

Queda prohibida la reproducción total o parcial de esta obra sin permiso escrito de los editores.

ESTETICA EN LOS PROGRAMAS

Cuando las aplicaciones de los ordenadores estaban restringidas al campo comercial, científico o estadístico, la preocupación de fabricantes y programadores se centraba en la necesidad de obtener unas salidas impresas que exigieran el menor tiempo de proceso posible.

La llegada del microordenador supuso un cambio radical en esta concepción: introducción del color, empleo habitual del sonido y proliferación de las pantallas como órganos de visualización de datos, con mayor protagonismo incluso que las impresoras.

El ordenador personal ya no es un equipo cuya orientación casi exclusiva es el tratamiento masivo de información, sino que su destino primario apunta a la satisfacción del usuario en las tareas más diversas, predominando las meramente recreativas y de aplicación doméstica; tareas en las cuales una representación visual atractiva, tanto en pantalla como impresora, adquiere vital importancia.

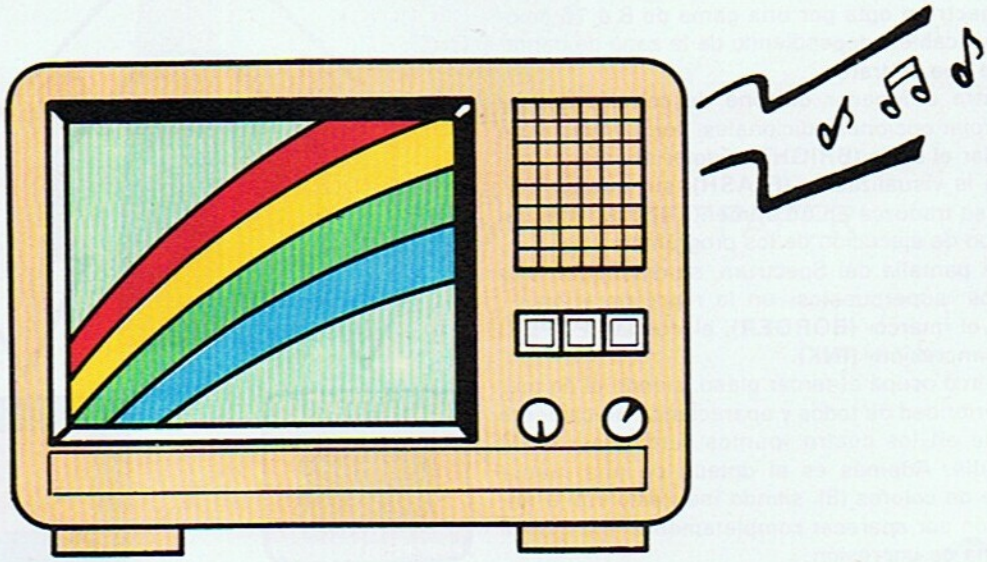
La llegada del microordenador trajo consigo la introducción del color, el sonido y las pantallas como órgano de visualización de datos.

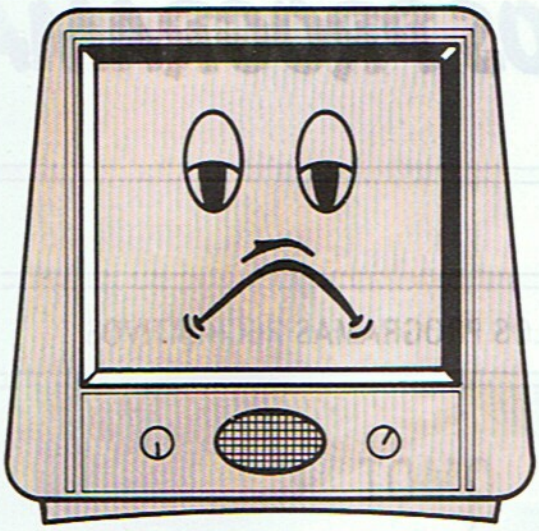
LOS PROGRAMAS RECREATIVOS

En los programas recreativos la estética desempeña un papel protagonista. Por muy notable que sea su calidad técnica o científica, la impresión que produce en el usuario un programa desprovisto de efectos de color y sonido no es comparable frente al atractivo estético de un juego de inferior calidad.

Las firmas comerciales, conscientes del fenómeno, decoran sus programas con todo tipo de sutilezas cromáticas y sonoras. Detalles que, sin embargo, el usuario pasa por alto la mayor parte de las veces al confeccionar sus propios programas. El resultado: programas que sin lugar a dudas cumplen con su propósito, pero cuyo aspecto y presentación resultan muy poco brillantes. El motivo: casi siempre por puro descuido, aunque bien es cierto que muchas veces la causa se encuentra en el desconocimiento de media docena de simples «trucos».

En los programas recreativos la estética desempeña un papel protagonista.





Normalmente el usuario confecciona sus programas sin estética, ni sonido por desconocimiento de media docena de simples «trucos».

¡!

La forma más habitual de emplear la técnica de menú, es situar una de estas rutinas al comienzo del programa, dando acceso a todas las opciones que lo integran.

*

Aunque los primeros ordenadores personales se limitaban al blanco y negro, casi todos los nuevos modelos incorporan la posibilidad del color.

*

Un aspecto de vital importancia, aun en las aplicaciones más triviales, radica en la correcta elección de los colores a emplear.

EL COLOR

Aunque los primeros ordenadores personales se limitaban al blanco y negro, casi todos los nuevos modelos incorporan la posibilidad del color. El Spectrum opta por una gama de 8 ó 16 colores aplicables, dependiendo de la zona de pantalla de que se trate.

Nuestro ordenador dispone de comandos para controlar opciones adicionales, por ejemplo, para regular el brillo **(BRIGHT)** o dotar de intermitencia a la visualización **(FLASH)**, sin que su empleo se traduzca en un aumento significativo del tiempo de ejecución de los programas.

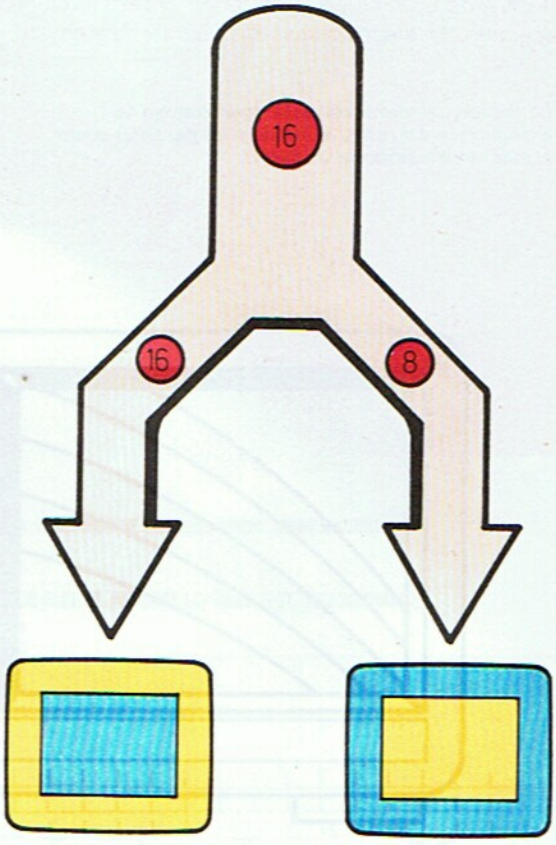
En la pantalla del Spectrum, se distinguen tres planos «superpuestos» en la representación visual: el «marco» **(BORDER)**, el «fondo» **(PAPER)** y la «impresión» **(INK)**.

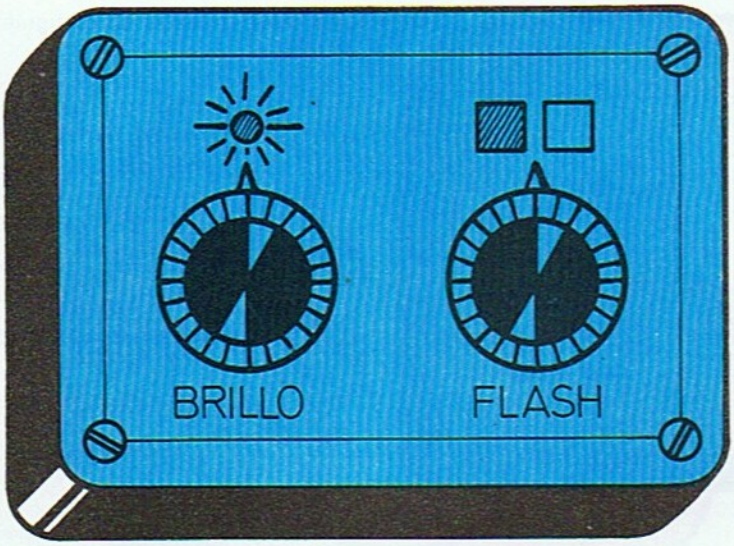
El marco ocupa el tercer plano, siendo el de menor prioridad de todos y apareciendo únicamente visible en los cuatro «puntos cardinales» de la pantalla. Además es el dotado de una menor gama de colores (8), siendo inaccesible a la impresión por aparecer completamente «exterior» a la zona de impresión.

No obstante, cumple una función de utilidad informativa en el momento de la gestión de grabación y carga de programas y series de *bytes*, anunciando en qué estado concreto se encuentra el proceso por medio de los cambios de color del contorno. En ocasiones, podemos hacer uso de esta posibilidad nosotros mismos en nuestros programas, e incluso conseguir bandas de diferentes colores que oscilen por el contorno de la pantalla con fines decorativos, como hemos podido ver en capítulos anteriores.

El fondo ocupa el segundo plano. Puede adoptar los 8 colores fundamentales válidos para el contorno y los 8 suplementarios haciendo uso de la adición de brillo. Además, puede incorporar también parpadeo, con lo que acapara el máximo de posibilidades de color. Como característica determinante podemos decir que sirve de fondo para la impresión de caracteres, delimitando perfecta-

El Spectrum opta por una gama de 8 ó 16 colores aplicables dependiendo de la zona de pantalla de que se trate.

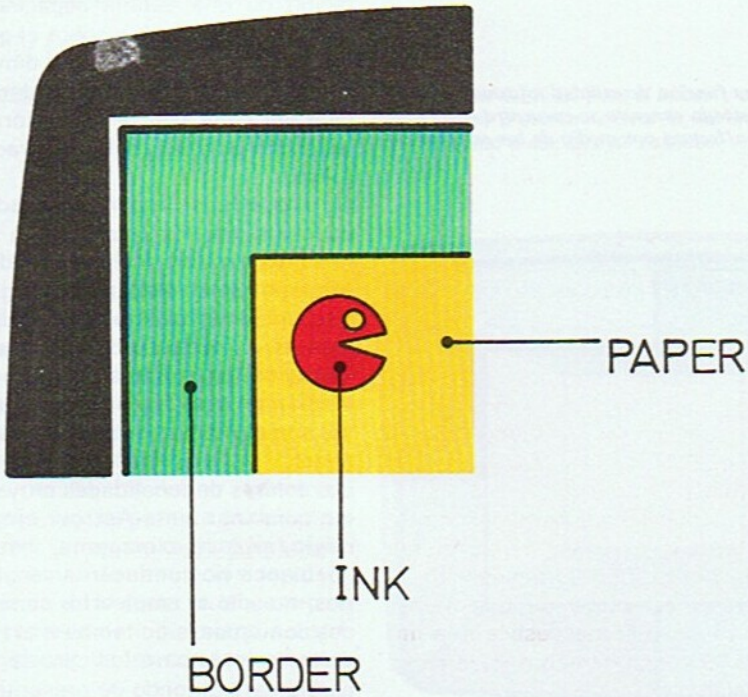




Nuestro ordenador dispone de comandos para controlar opciones adicionales como el brillo y la intermitencia o **FLASH.**

mente el contorno imprimible de la pantalla al hacer frontera con el tercer plano.
Dentro de una variación tonal no excesivamente acusada, la elección de un mismo color básico para el marco y el fondo o zona útil de la pantalla crea una ilusión de mayor tamaño y uniformidad. Sin embargo, en otras ocasiones deseaemos precisamente lo contrario. Es en este segundo caso, cuando coinciden colores diferentes para

En la pantalla del Spectrum, se distinguen tres planos «superpuestos»: el marco, el fondo y la impresión.



¡!

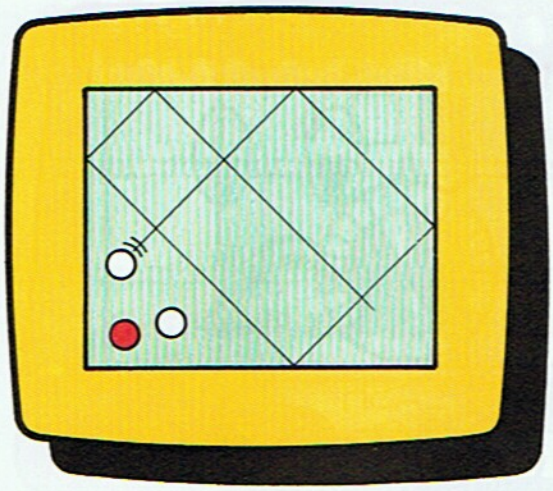
Aunque los juegos redefinidos de caracteres ocupan parte de la RAM, no suelen presentar problemas de memoria, puesto que dejan suficiente espacio para otras tareas.

*

Una posibilidad m[illegible] tener en cuenta es[illegible] de simultanear dos t[illegible]pos de letra.

*

En el próximo capítulo de la sección PROGRAMA, podremos encontrar la información suficiente para adoptar hasta cuatro juegos de caracteres distintos al original de nuestro Spectrum.



El marco es inaccesible a la impresión por aparecer completamente «exterior» a la zona con dicha función.

el marco y el fondo, donde podemos añadir una nota de originalidad al programa incluyendo una fina línea trazada bordeando el contorno de la zona de impresión, para delimitar mejor la zona de «incertidumbre» de los dos colores, o incluso reforzarla para dar idea de volumen. Podemos ver una breve muestra en el siguiente programa de ejemplo:

```
  10 REM - DELIMITADOR CONTORNO (C) 1985
 LOPEZ MARTINEZ
  20 BORDER 7: PAPER 7: CLS
  30 PRINT AT 21,0;"BORDER","PAPER"
  40 INPUT B,P: IF B<0 OR B>8 OR P<0 OR
P>8 THEN  BEEP .25,0: GO TO 40
  50 BORDER B: PAPER P: INK 9: CLS
  60 PRINT #0;TAB 8;"PULSE UNA TECLA": P
AUSE 0
  70 DRAW 255,0: DRAW 0,175: DRAW -255,0
: DRAW 0,-175
  80 PAUSE 0
  90 PLOT 1,1: DRAW 253,0: DRAW 0,174
 100 PAUSE 0
 110 PLOT 1,2: DRAW 252,0: DRAW 0,173
 120 PAUSE 0
 130 GO TO 20
```

Por último llegamos al primer plano: el ocupado por la impresión propiamente dicha **(INK)**, siempre superpuesto forzosamente sobre el fondo y vinculado a las mismas posibilidades de color que para éste.

Resulta difícil dar unas normas generales, puesto que las preferencias estéticas dependen en gran medida del gusto personal de cada programador. Sin embargo, a modo orientativo, pueden servirnos de ayuda algunas puntualizaciones.

Es conveniente que los colores se adapten en la medida de lo posible a la realidad. Por ejemplo, en una batalla naval el cielo debería ser cián y el mar azul; reservando para los barcos, submarinos, torpedos y cargas de profundidad colores que destaquen sobre el elegido para el fondo de la pantalla en el que se desplazan. En la simulación de una batalla espacial es conveniente elegir un fondo negro sobre el que destaquen las estrellas, representadas por diminutos puntos luminosos de color blanco. Además, la combinación del color con la definición de caracteres o unidades gráficas realza el atractivo de cualquier juego.

Otro aspecto de importancia radica en elegir cuidadosamente los colores, aún en las aplicaciones más triviales. Incluso cuando sólo se trata de imprimir textos o editar un programa, se observa que un fondo oscuro (preferiblemente azul con los textos escritos en blanco), resulta menos molesto y fatigoso para la vista que la tradicional impresión de caracteres en negro sobre fondo blanco, a la que estamos sujetos al conectar el ordenador.

Los colores de tonalidades muy próximas no suelen combinar bien. Así, por ejemplo, las parejas negro/azul, rojo/magenta, verde/cián o amarillo/blanco no conducen a resultados satisfactorios; no sólo al emplearlos como tonos para fondos contiguos, sino también al hacerlo como uno de los colores para los caracteres del texto y su pareja para el fondo de presentación.

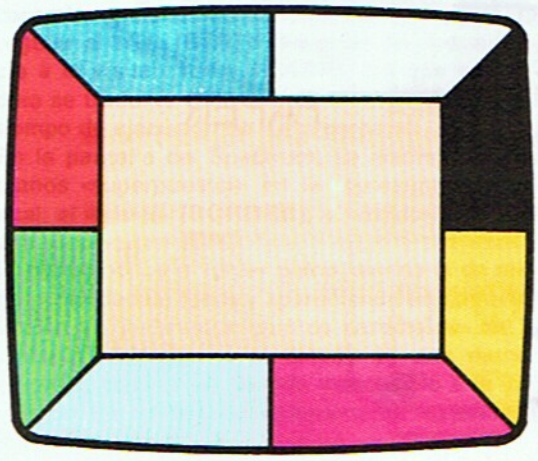
¡!

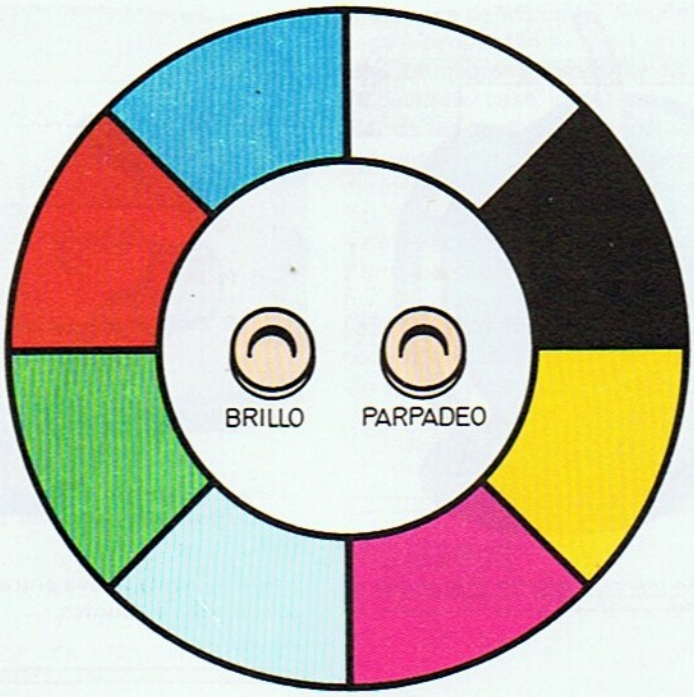
En el caso del menú por número de opción, la depuración de entrada por programa es más sencilla; no obstante, con el sistema de inicial se consiguen resultados más próximos al entendimiento con el operador.

*

El ordenador personal ya no es un equipo cuya orientación casi exclusiva es el tratamiento masivo de la información.

El marco cumple una función de utilidad informativa, anunciando en qué estado concreto se encuentra el proceso de grabación/lectura por medio de los cambios de color del contorno.





El fondo puede adoptar los 8 colores fundamentales, y los 8 suplementarios haciendo uso del brillo y el parpadeo.

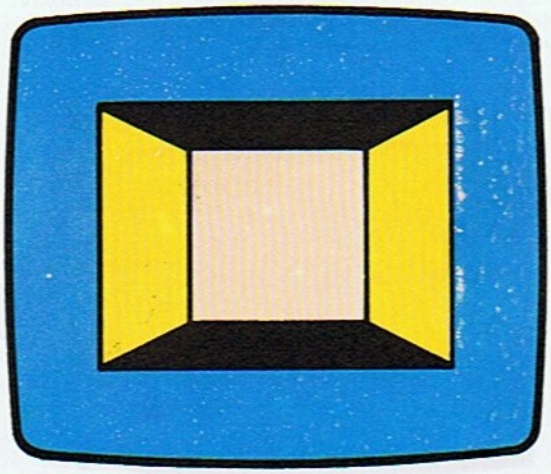
Otro ejemplo ilustrativo podemos encontrarlo en los juegos de ajedrez. Aunque la atención esté centrada en el tablero de juego, ésta puede dirigirse en cualquier momento a la «ventana» de la pantalla que representa los mensajes relativos a

ORGANIZACION DE LA PANTALLA

La elección del color para el marco y el fondo es primordial, si deseamos crear la ilusión de mayor tamaño, uniformidad o dar idea de volumen.

La correcta distribución de «masas» en la pantalla es otra de las técnicas que influyen en gran medida en el logro de una presentación más atractiva. Hay que procurar que la acción transcurra en el centro, dejando las líneas superiores e inferiores de la misma para la representación de indicadores para el operador.

En algunos casos resulta útil definir «ventanas» dentro de la pantalla, lo que permite simultanear dos o más acciones. Un ejemplo claro lo encontramos en los programas de simulación de vuelo, con la representación de la pista en el centro de la pantalla y los cuadros de instrumentos distribuidos alrededor, dentro de las correspondientes «ventanas».



¡!

La técnica de selección por menú se puede considerar casi patrimonio de los programas de gestión, por ser éstos los que acaparan mayor complejidad en cuanto a diversidad de tareas a realizar.

*

Los menús se pueden confeccionar según dos técnicas: selección en base al número de opción y por la inicial de la frase que califica la opción.

*

Con la opción de parpadeo lo que realmente se consigue es una impresión alternativa positivo/negativo de dudoso contenido estético.

*

La estética del parpadeo conseguido mediante **FLASH** tiene la ventaja adicional de representar un considerable ahorro de memoria con relación a otros sistemas de efectos parecidos.

¡!

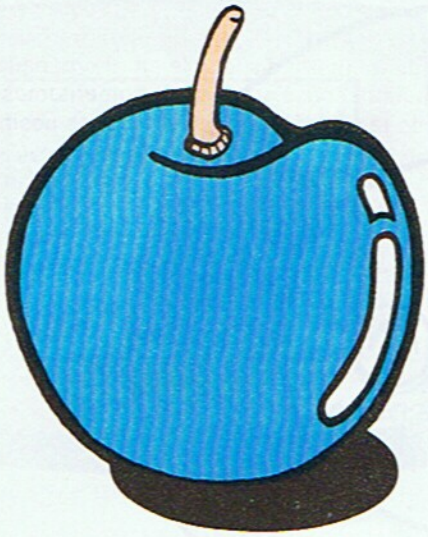
Las firmas comerciales conscientes del atractivo estético de un juego, decoran sus programas con todo tipo de sutilezas cromáticas y sonoras.

*

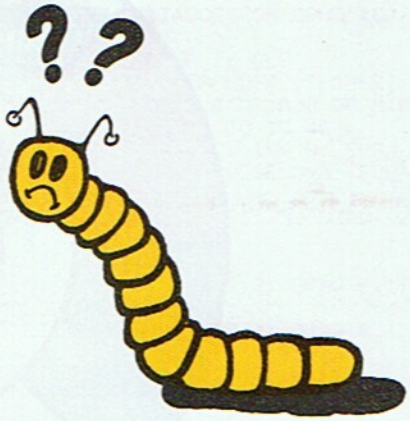
El fondo delimita las fronteras de la zona imprimible de la pantalla.

*

El primer plano (INK) se encuentra forzosamente sobre el fondo y vinculado a las posibilidades de color de éste.



Es conveniente que los colores se adapten en la medida de lo posible a la realidad.



la validez del movimiento realizado, o a la «ventana» que relaciona los movimientos ejecutados con anterioridad.

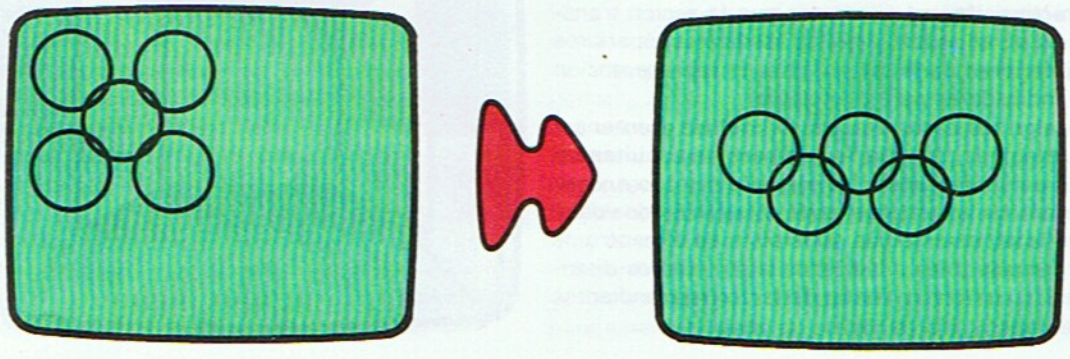
Por último, los programas de gestión pueden ser buenos receptores de esta técnica, permitiendo así un *scrolling* (desplazamiento vertical) parcial de la pantalla. Esta técnica sería especialmente útil en entradas masivas de datos (Contabilidad, Almacén, etc...), manteniéndose fijas las primeras líneas de pantalla para contener las cabeceras de la petición de datos, y desplazando hacia arriba por cada nueva entrada el bloque de líneas de «detalle» o apuntes.

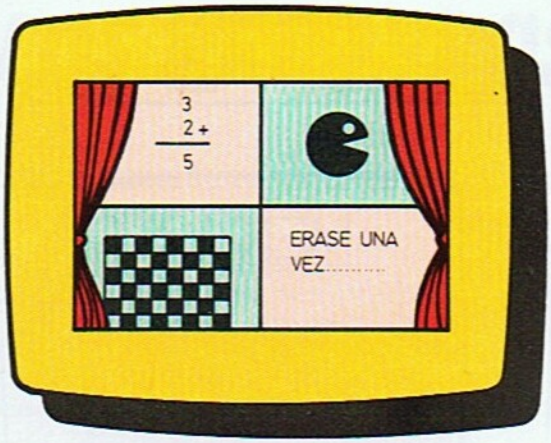
MENSAJES AL OPERADOR

Otro capítulo importante es el de los mensajes orientativos para el operador. Una buena técnica de aprovechamiento puede consistir en emplear para ello la línea 23 de la pantalla por medio de la expresión **PRINT #0**, con textos centrados por la función **TAB**. Estos mensajes han de borrarse antes de ser reemplazados por otros nuevos con **INPUT ""**, de la forma indicada en el programa ejemplo anterior.

Sin embargo, podemos introducir una mejora para los más exigentes, que acapara mayor atención, como la expuesta en el siguiente programa de ejemplo:

La correcta distribución de «masas» en la pantalla es otra de las técnicas que influyen en gran medida en el logro de una presentación más atractiva.





En algunos casos resulta útil definir «ventanas» dentro de la pantalla, lo que permite simultanear dos o más acciones.

```
  10 REM - MENSAJES (C) 1985 LOPEZ MARTI
NEZ
  20 INPUT "MENSAJE:"; LINE X$
  30 FOR I=1 TO LEN X$
  40 PRINT )0;X$(I);" ";CHR$ 8;: BEEP .
025,25
  50 NEXT I
  60 PRINT )0;" ": BEEP .25,10: PAUSE 0:
 GO TO 20
```

En este programa hemos incluido sonido y acción. De hecho, esta forma de escritura es la empleada comúnmente para la representación en pantalla de textos largos. Esto es debido a que la salida a pantalla del ordenador es mucho más rápida que la lectura que podemos hacer de las frases, por lo que se hace necesario, además de por puro afán estético, para retardar de alguna manera la impresión.

LOS JUEGOS DE CARACTERES

Hemos visto en capítulos anteriores cómo pueden definirse nuevos juegos de caracteres en RAM. Si bien es cierto que la labor de generar un juego original es bastante árdua, no lo es menos que podemos sacar gran partido de las horas empleadas en este cometido, puesto que podemos cargar en veces sucesivas en memoria el juego definido en forma de *bytes*.
Aunque los juegos redefinidos ocupan parte de RAM no suelen representar problema, ya que en la mayoría de los casos ningún programa BASIC llega a ocupar tan siquiera una mínima parte del total de memoria disponible en el ordenador.
En cualquier caso, y esto ya lo comentamos al hablar de las Variables del Sistema, es posible hacer una redefinición parcial que incluya únicamente los números y caracteres en mayúsculas. Por último, y esta solución es únicamente recomendable para programas en los cuales vayamos a emplear masivamente la representación de textos en la pantalla, podemos simultanear dos tipos de letra definidos al mismo tiempo. Para ello, podemos fijar un juego de caracteres en «cursiva» para las letras minúsculas y otro compatible con éste, aunque utilizable por su parte sólo con las mayúsculas y los números.

LOS MENUS

Se conoce comúnmente por «menú» al sistema de elección del operador por indicación directa de

Se conoce comúnmente por «menú» al sistema de elección del operador por indicación directa de un número o letra.

¡!

En ocasiones podemos conseguir bandas de diferentes colores que oscilen por el contorno de la pantalla con fines decorativos.

¡!

Una buena técnica de aprovechamiento del espacio disponible en la pantalla, es emplear la línea 23 (reservada al Sistema) para la edición de mensajes con destino al operador.

*

El marco se hace bien visible únicamente en los cuatro puntos cardinales de la pantalla.

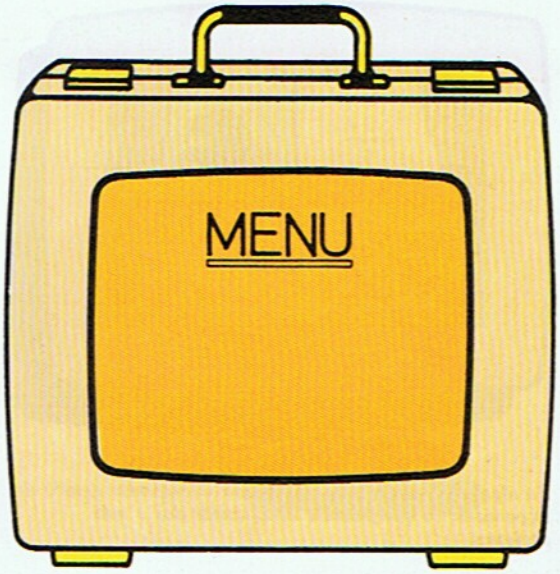
La técnica de selección por menú es casi patrimonio de los programas de gestión.

un número o letra, a cuyo margen se encuentra un texto explicativo de la labor que tendrá lugar al seleccionarse la opción. La forma más simple es que un único menú dé acceso a todas las facetas de un programa, sin embargo, puede darse el caso de que un menú «general» dé paso a otros, constituyendo una estructura jerarquizada de procedimiento.

La técnica de selección por menú es casi patrimonio de los programas de gestión, por ser éstos los que acaparan mayor complejidad en cuanto a diversidad de tareas a realizar, dentro de un área específica de la administración de un negocio como puede ser la Contabilidad, o la Gestión Comercial.

Existen dos técnicas de tratamiento. La primera de ellas, la más simple, consiste en numerar las opciones y solicitar que se teclee un número. La ventaja del sistema reside en la facilidad de depuración, ya que cualquier número inferior al mínimo o superior al máximo impresos, será rechazado automáticamente.

El segundo sistema ofrece más dificultad a la hora de la depuración. Por ello, suele recurrirse a él cuando se trata de seleccionar entre un número bajo de opciones. Consiste en calificar cada



opción con la inicial de la primera palabra, o de la más significativa si existen repetidas.

EL ATRIBUTO FLASH

Una de las aplicaciones de la impresión con «parpadeo» o **FLASH** es la representación de mensajes, sobre todo de los que requieren llamar más la atención por ser necesario realizar alguna operación manual (colocar papel en la impresora, etc...).

Sin embargo, a pesar de poseer el Spectrum la posibilidad de impresión en parpadeo, ésta no resulta del todo satisfactoria, pues se reduce a intercambiar constantemente los atributos **PAPER** e **INK** de las posiciones de pantalla afectadas, con lo cual realmente se consigue una impresión en negativo y positivo, alternativamente.

La razón de este estado de cosas es, sin duda, el ahorro de memoria, ya que para efectuar esta alteración de atributos no se precisa de memoria suplementaria. Como solución para los más exigentes proponemos hacer el tratamiento de estos mensajes de forma manual, representando primero el mensaje en pantalla en la posición adecuada para seguidamente borrarlo.

Repitiendo el proceso antes descrito, e incorporando un zumbido entre aparición y desaparición del texto, podemos conseguir un mensaje de aviso similar al que podemos encontrar en ordenadores más grandes y con mayor calidad estética.

Una de las aplicaciones de la impresión con «parpadeo» es la representación de mensajes que requieren llamar más la atención.

LOS PRIMEROS PASOS

Los próximos doce capítulos, es decir, todos los que componen el cuarto tomo de la presente obra dentro de la sección «TU SPECTRUM», estarán íntegramente dedicados al código máquina o lenguaje máquina del microprocesador Z80.

Por tratarse de un lenguaje completamente diferente al BASIC, al menos en el formato de sus instrucciones, potencia, posibilidades y técnicas de construcción de programas, en principio, no es necesario conocer a fondo éste último, pues parten de bases distintas, aunque ambos persiguen un objetivo común: informar claramente al Spectrum, cuál es el trabajo que debe ejecutar.

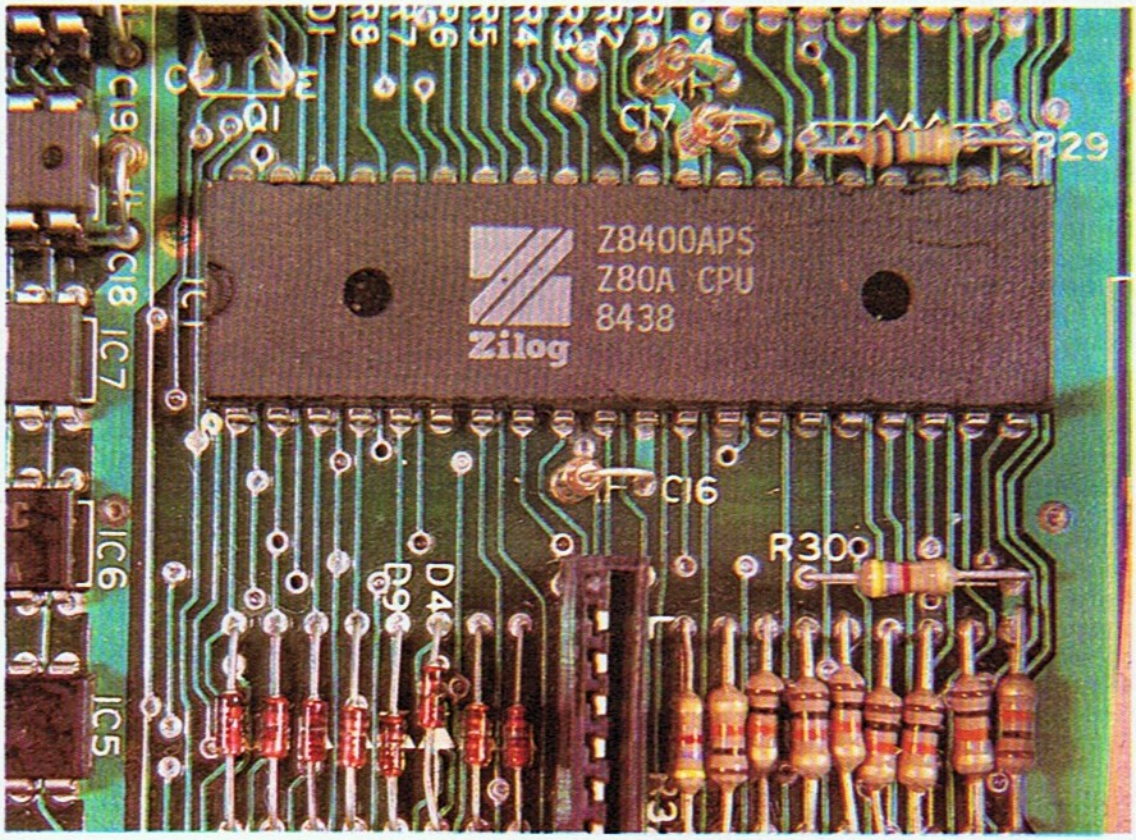
El lenguaje máquina es lo más próximo a la forma de hablar de nuestro micro. Por ello, su conocimiento y comprensión nos descubrirá sus auténticas posibilidades y la manera de funcionar de nuestro pequeño cerebro electrónico.

Tres capítulos fueron dedicados en su momento como introducción a los temas que a continuación trataremos. Volvamos a repasarlos, pues en ellos encontraremos los conceptos previos y las indicaciones necesarias para convertir el lenguaje máquina en una técnica de programación tan fácil de asimilar como lo pudo ser el BASIC.

No desesperemos si el aluvión de nuevos conceptos desborda nuestras previsiones. En poco tiempo, y con alguna práctica seremos capaces de manejarlos a nuestro antojo. Todos los programas y ejemplos estarán en forma clara y debidamente documentados para que podamos detectar cualquier posible equivocación de carga.

No olvidemos que los errores de programación en C/M no pueden estropear nuestro ordenador.

El microprocesador Z 80 A es el verdadero «cerebro» de nuestro micro.



¡!

El contenido de esta sección durante el presente volumen estará íntegramente dedicado al estudio del código máquina.

*

No es preciso conocer a fondo el BASIC para aprender C/M.

Todo lo más, el «desastre» provocará la pérdida del control sobre nuestro micro, y el problema quedará solucionado desconectando la alimentación o pulsando el botón RESET. Por ello, antes de ejecutarlo conviene grabarlo como seguro del tiempo empleado introduciéndolo a través del teclado.

LOS PROGRAMAS EN C/M

Como ocurre en el caso de los escritos en BASIC, los programas en C/M deben estar almacenados en la memoria del ordenador antes de ser ejecutados. No olvidemos que la ROM, a fin de cuentas, es también una memoria aunque no podamos alterarla, y por tanto, tendremos la posibilidad, si lo creemos necesario, de leer los 16 Kbytes en C/M almacenados allí, y utilizarlos en nuestros programas.

Un programa en C/M es una secuencia de instrucciones encargadas de procesar ciertos datos, las cuales permiten realizar al ordenador una serie de operaciones que completen una determinada tarea, pero con una diferencia sustancial respecto a otros lenguajes como el BASIC: son directamente interpretadas por el ordenador sin necesidad de emplear un traductor intermedio. De ahí, su extremada velocidad de proceso y a la par, la complejidad de su construcción.

De lo anterior deducimos que en la memoria de nuestro Spectrum (RAM o ROM) podemos encontrar tanto instrucciones como datos, pero almacenados no de cualquier manera, sino en forma de dígitos binarios o bits (0 y 1) correspondientes, como sabemos, a dos niveles eléctricos diferentes, agrupados de ocho en ocho en unidades mayores denominadas bytes.

Pero ¿cómo se ejecuta un programa? Pues bien, el Z80-A, microprocesador del Spectrum, lee de la memoria una instrucción, analiza el byte o los bytes que la componen y en función de su contenido, fielmente realiza la operación indicada y así sucesivamente hasta completar su trabajo.

Como vemos, el verdadero «cerebro» en todas las tareas que efectúa nuestro ordenador es su microprocesador. Ya hemos comentado en esta misma sección que físicamente está constituido por miles de pequeños componentes electrónicos encapsulados en una única pastilla o chip.

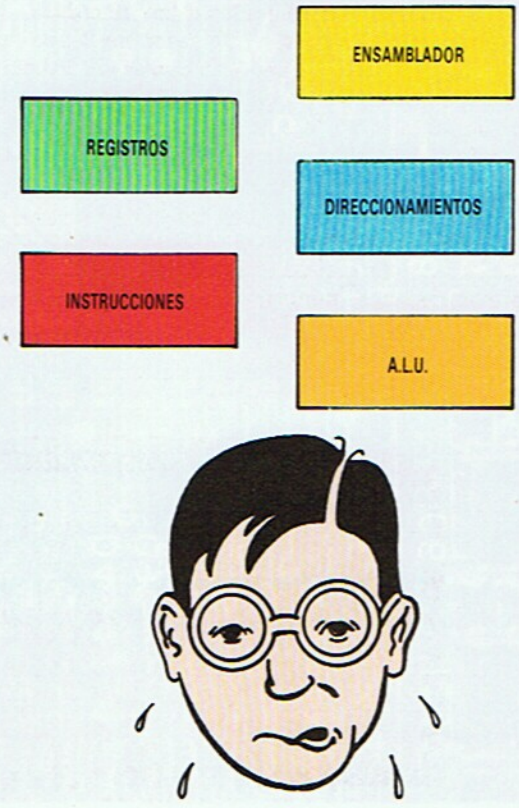
El cómo funcionan y su interconexión eléctrica es algo que para el estudio del C/M no nos interesa. Sin embargo, sí es importante conocer cuales son los componentes lógicos alojados dentro de este chip, pues son la razón fundamental que justifica el gran número de micros que lo incorporan.

Antes de seguir una aclaración: el sufijo A indica que se trata de una versión mejorada del antiguo modelo, la cual es capaz de funcionar mucho más rápidamente que su predecesor (ver el capítulo titulado EL INTERIOR DE TU SPECTRUM), pero siendo sus juegos de instrucciones totalmente equivalentes. Por ello, siempre que nos refiramos a él en adelante, lo haremos bajo la denominación Z80.

LA CPU

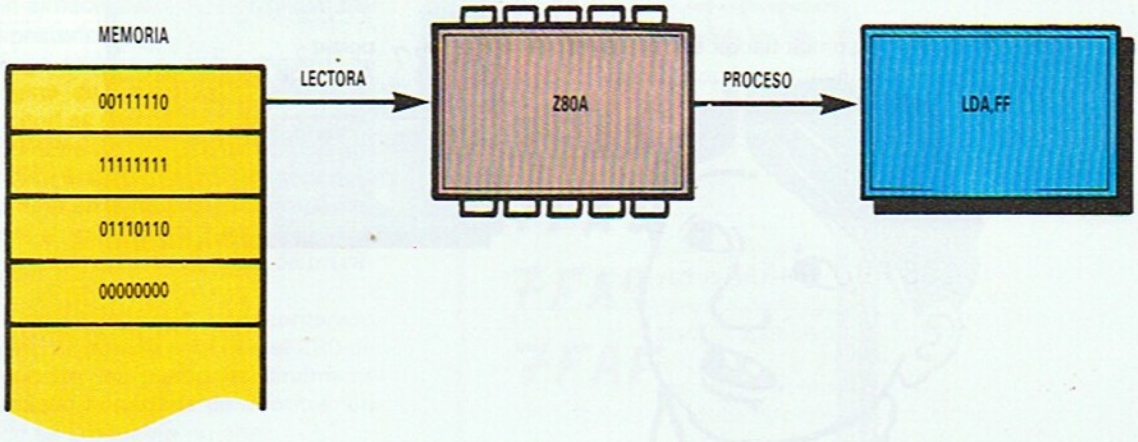
En la unidad central de proceso de nuestro Spectrum podemos encontrar los siguientes tres bloques lógicos principales:

No desesperemos si el aluvión de nuevos conceptos desborda nuestras previsiones.



¡!

Es muy importante repasar los capítulos dedicados anteriormente al lenguaje máquina.



El Z80 A lee de la memoria una instrucción y ejecuta la operación indicada por ésta.

LOS REGISTROS

— Los buses.
— La unidad aritmética y lógica (ALU).
— Los registros.

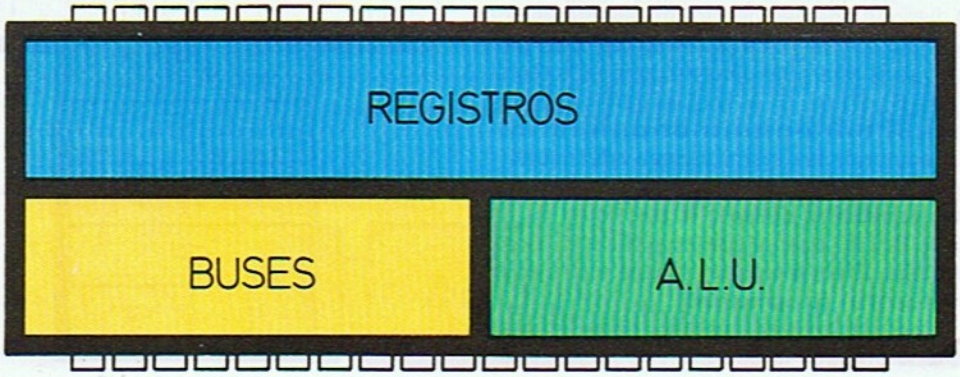
Como ocurre en el resto del ordenador, la información que circula a través de los diferentes componentes del interior del Z80 lo hace a través de los buses o conjunto de conductores eléctricos, uno por bit, habilitados a tan efecto.
Son tres los presentes: el de datos de 8 bits, el de control de 13 bits para sincronizar la CPU con los dispositivos exteriores a ella, y el de direcciones de 16 bits, por lo cual el Z80 es capaz de direccionar 2↑16=65536 (64 Kbytes) posiciones diferentes de memoria de un byte cada una.
La ALU *(Arithmetic Logic Unit)* es la calculadora interna de la CPU. Las operaciones que puede ejecutar son las más básicas: sumas, restas, comparaciones, incrementar o decrementar una cantidad en uno, pero no multiplica ni divide.

Un registro es un circuito del interior del Z80 utilizado como almacenamiento intermedio de información. Los registros disponibles en este microprocesador podemos clasificarlos en los siguientes grupos:

— Registros de uso general.
— Registros de uso específico.
— Registros de control.

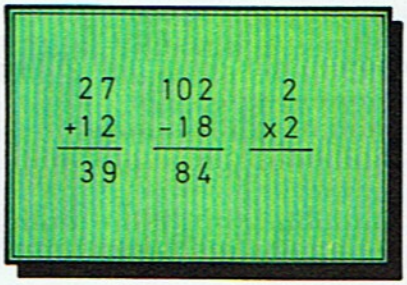
Los registros de uso general son utilizados a voluntad por el usuario e incluyen un acumulador A, y los registros B, C, D, E, H y L. Cada uno de ellos tiene una capacidad de 8 bits y pueden em-

Registros del Z80.



Un registro es un circuito del interior del Z80 usado como almacenamiento temporal de información.

IX e IY son denominados, respectivamente, registros índice X e Y.



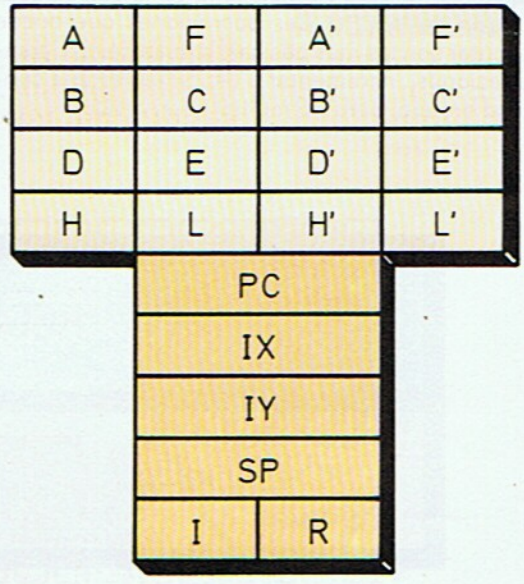
La unidad aritmética y lógica no multiplica ni divide.

plearse solos o por parejas, BC, DE y HL (constituyéndose en unidades de 16 bits).
También está presente un registro de indicadores F (Flags) con capacidad para 8 bits, siendo su contenido variable según el resultado de determinadas operaciones que más adelante discutiremos en detalle.
El significado de cada uno de los bits que lo componen está íntimamente relacionado con el funcionamiento interno del microprocesador, siendo su conocimiento determinante para la correcta programación. Señalar también, que algunas instrucciones utilizan la pareja AF para efectuar su cometido.
Como comprobaremos en su momento, entre los registros de uso general, el microprocesador utiliza algunos de forma preferente, tales como el acumulador (A) o la pareja HL, pues determinadas operaciones sólo pueden ser ejecutadas con su concurso.
El Z80 dispone además de otro grupo equivalente denominado ARS (Alternate Register Set, juego de registros alternativo) formado por los registros A', F', B', C', D', E', H' y L', el cual puede intercambiarse con el anterior en cualquier momento ejecutando las instrucciones adecuadas.
Los de uso específico son registros que cumplen misiones especiales durante la ejecución de un programa. El Z80 cuenta con cuatro de 16 bits y dos de 8 bits. Efectuemos un repaso sobre ellos y veamos cuál es su cometido:
— PC *(Program Counter,* contador de programa). En él queda anotada la dirección de memoria de la instrucción en curso durante la ejecución del programa. Su capacidad son 16 bits.
— IX e IY son dos registros con capacidad cada uno de 16 bits, denominados registros INDICE X e Y, respectivamente. Son manejados por el microprocesador durante las operaciones de direccionamiento indexado (ya lo discutiremos en su momento), siendo especialmente útiles para el acceso a tablas de datos.
— SP *(Stack Pointer,* puntero de la pila) es un registro de 16 bits cuya función consiste en indicar al microprocesador dónde se encuentra una zona

¡!

El orden en que el microprocesador encuentra la información determina el tipo de que se trata.

Z80: bloques lógicos.



especial de la memoria RAM denominada STACK o pila, donde son almacenados determinados datos para su uso posterior.

— I *(Interrupt page address register,* registro de dirección de página de interrupción) empleado cuando una solicitud es efectuada al microprocesador por un periférico de alta prioridad, y al objeto de ser atendida ésta, se rompe la secuencia normal del programa en ejecución, continuándose posteriormente el trabajo desde el punto en el cual se produjo la interrupción. Se trata de un registro de 8 bits.

— R *(memory Refresh register),* cuyo contenido es cíclicamente incrementado para que el Z80 dé la orden de regenerar las memorias dinámicas del Sistema, evitando la pérdida de información que de otro modo se produciría en ellas.

Además de todos los anteriores, la CPU de nuestro Spectrum cuenta con otros registros no accesibles por el programador como, por ejemplo, el IR *(Instruction Register,* registro de instrucciones), el cual contiene la instrucción que va a ejecutarse mientras es decodificada (interpretada) por el microprocesador, o registros de control para uso temporal durante las operaciones que se efectúen.

El el registro PC queda anotada la dirección de memoria donde el microprocesador encontrará la siguiente instrucción a procesar.

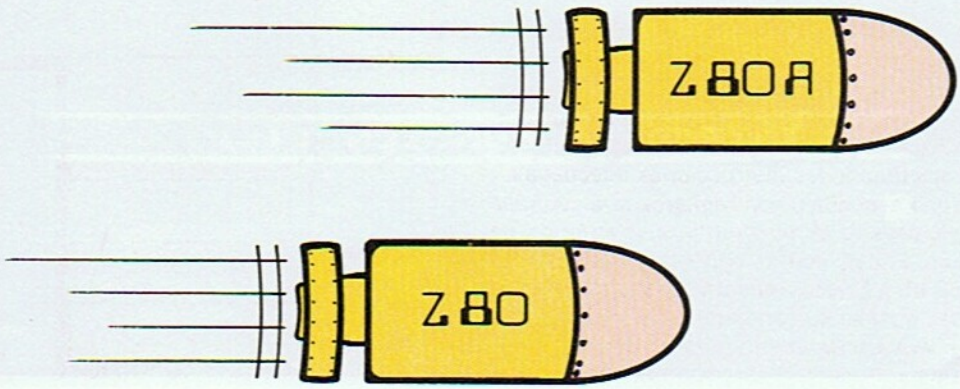
EJECUTANDO PROGRAMAS

Como hemos expuesto anteriormente, la información necesaria para que un programa en C/M pueda ejecutarse (datos+instrucciones) debe encontrarse previamente almacenada en la memoria del ordenador.

Algunas instrucciones ocupan un único byte, pero también las hay de dos, tres y cuatro bytes, siendo su contenido variable en función de la tarea que realizan. Entonces, ¿cómo reconoce el Z80 los diferentes tipos de información presentes en la memoria sin dar lugar a ambigüedades? Es precisamente el orden en que el microprocesador la encuentra el que determina el tipo de que se trata. Por ésto, los formatos de las instrucciones están sujetos a determinadas normas que los hacen inconfundibles para la CPU, y les prestaremos especial atención para ceñirnos a las «leyes» de programación del lenguaje máquina del Z80.

En las tablas del próximo capítulo podremos encontrar el juego completo de instrucciones del microprocesador Z80, sus códigos de operación y la representación simbólica (mnemónicos) equivalente.

El Z80 A es una versión mejorada del anterior modelo.



Los registros de uso general son utilizados a voluntad por el usuario.

*

Se conoce bajo las siglas ARS *(Alternate Register Set)* al grupo de registros alternativo.

TRIGONOMETRIA

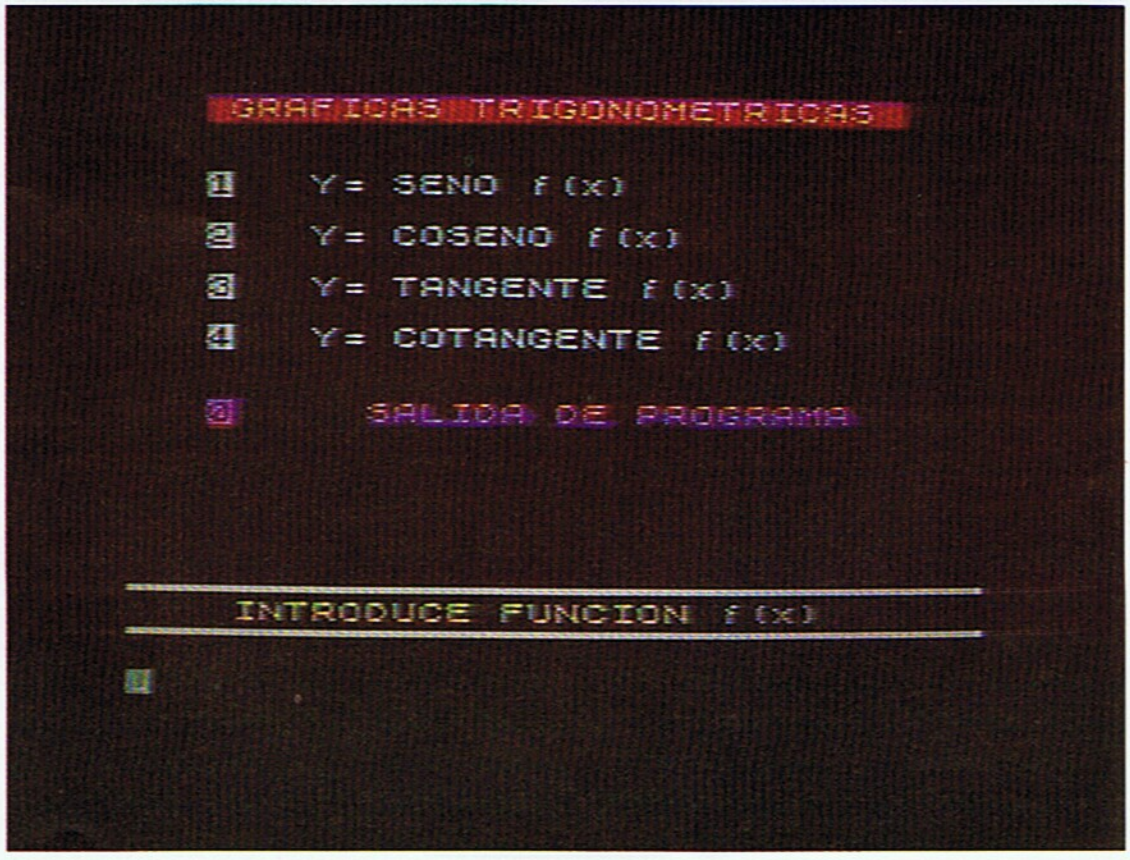
TRIGONOMETRÍA es un sencillo programa que ha sido especialmente desarrollado para todos aquellos estudiantes que se hayan introducido recientemente en el intrincado, y a la vez bonito mundo de esa rama de las matemáticas. Este programa ahorrará además un tiempo precioso en la ejecución de gráficas trigonométricas simples.

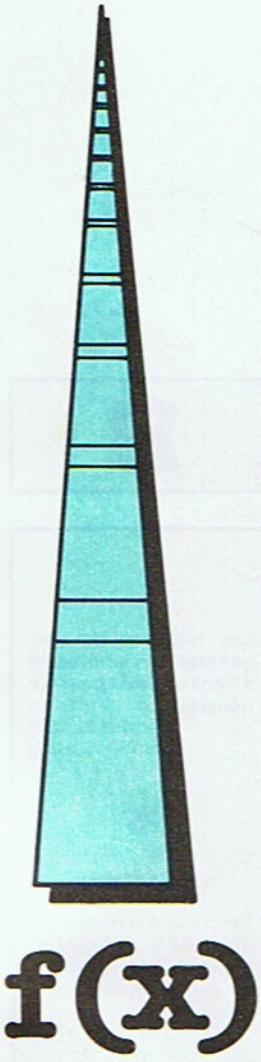
EL PROGRAMA

Una vez que hayamos introducido y ejecutado correctamente el programa, surgirá en la pantalla

TRIGONOMETRIA es un sencillo programa desarrollado para todo aquel que se haya introducido recientemente en esta rama de las matemáticas.

Para acceder a cualquiera de las cinco opciones no tenemos más que pulsar la tecla que corresponde al número de opción del gráfico que deseamos realizar.



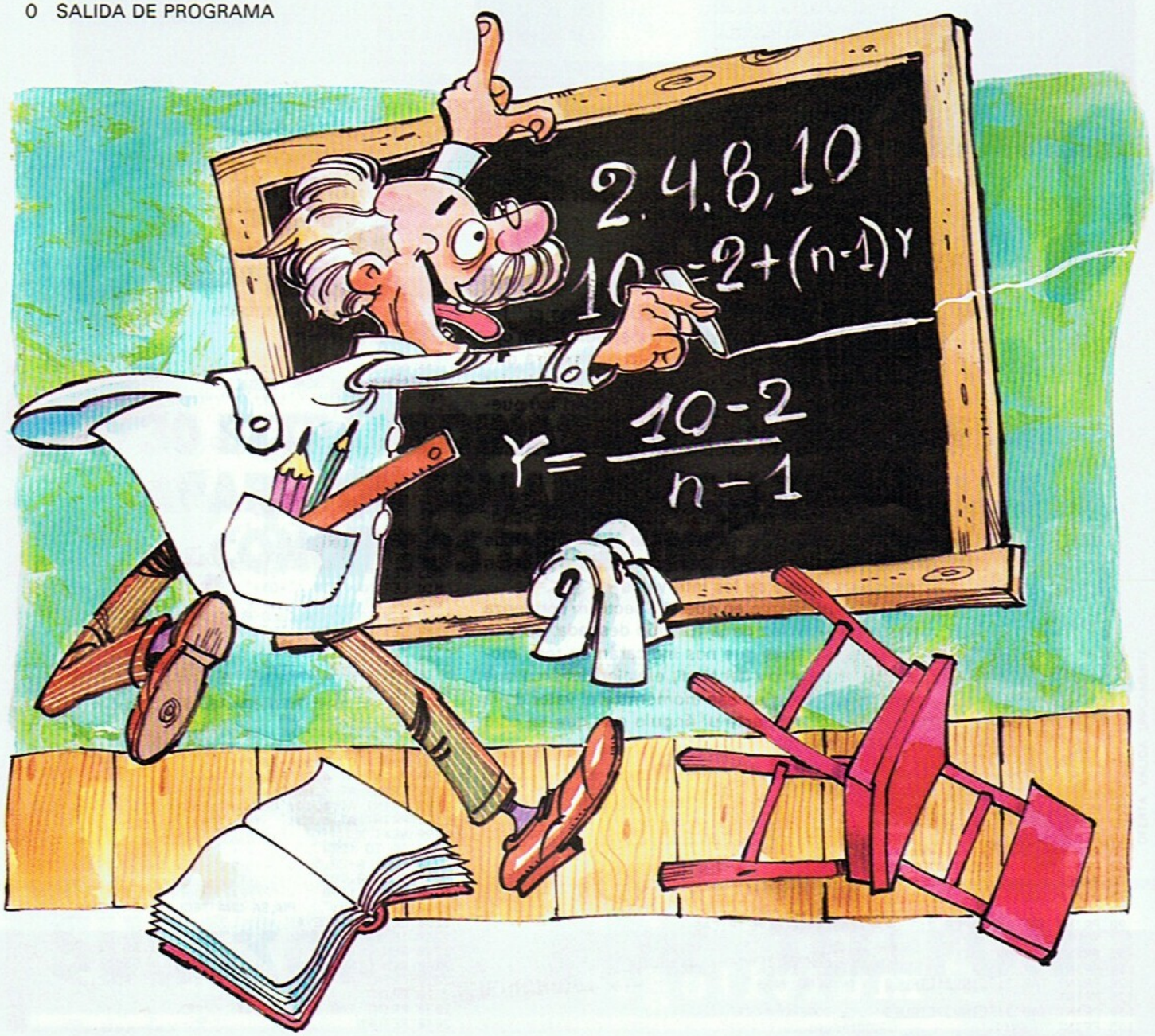


La función f(x) puede ser cualquier expresión matemática siempre y cuando no existan valores de X que proporcionen resultados infinitos.

de nuestro monitor o televisor un pequeño menú de 5 opciones:

1 Y = SENO f(x)
2 Y = COSENO f(x)
3 Y = TANGENTE f(x)
4 Y = COTANGENTE f(x)
0 SALIDA DE PROGRAMA

Para acceder a cualquiera de las cinco opciones no tenemos más que pulsar la tecla que corresponda al número de opción, del gráfico que deseamos realizar. Así pues, si tenemos que efec-



La representación de la función se efectúa en la parte inferior de la pantalla, quedando la superior reservada a la impresión de diversos datos de interés.

tuar la gráfica del SENO f(x), pulsaremos el "1". Inmediatamente después de haber pulsado la opción deseada, nuestro Spectrum nos pedirá que introduzcamos la función que queremos calcular y representar en la pantalla. La función f(x) puede ser cualquier expresión matemática siempre y cuando no existan valores de X que hagan la función infinita.

Otra pequeña restricción al uso de este programa, viene dada porque éste sólo puede representar aquellas funciones en donde todos o parte de sus valores (dependientes del parámetro X) estén definidos dentro de los intervalos –1<=x<=1.

Desde el momento en que el Spectrum comienza la representación de la función deseada, surgen unos marcadores que nos indicarán en todo momento la posición del pixel, el valor del arco que está en vigencia en ese momento y el valor de la función con respecto al ángulo con que se está operando.

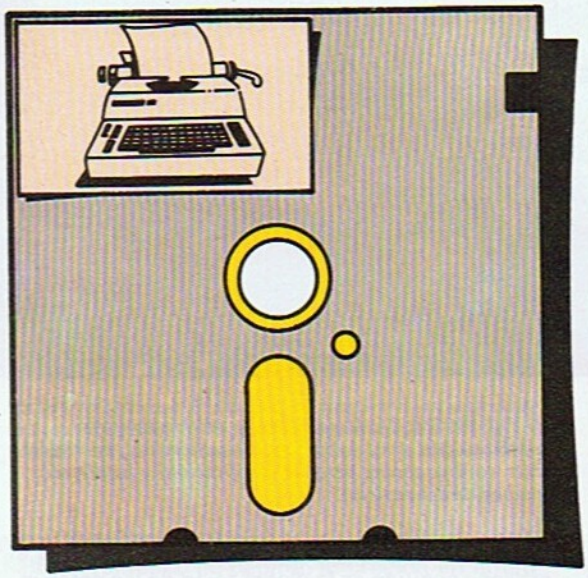
```
 10 REM ************************
 20 REM * J.M.MAYORAL SERRANO *
 30 REM ************************
 40 REM * PROFESOR   (C) 1985 *
 50 REM ************************
 60 REM [INV] PRESENTACION MENU [TRUE]
 70 PAPER 0: INK 9: BORDER 0
 75 DIM P(201): DIM Y(201)
 80 CLS
 90 PRINT TAB 3; PAPER 2;" GRAFICAS TRIGONOMETRICAS

 95 PRINT
110 PRINT
120 PRINT TAB 3;"[INV]1[TRUE]   Y= SENO f(x)"
125 PRINT
130 PRINT TAB 3;"[INV]2[TRUE]   Y= COSENO f(x)"
135 PRINT
140 PRINT TAB 3;"[INV]3[TRUE]   Y= TANGENTE f(x)"
145 PRINT
150 PRINT TAB 3;"[INV]4[TRUE]   Y= COTANGENTE f(x)"
155 PRINT
170 PRINT
177 PRINT  BRIGHT 1; INK 3;TAB 3;"[INV]0[TRUE]     S
ALIDA DE PROGRAMA"
180 PRINT  OVER 1;AT 18,3; INK 5;" PULSA LA OPCION D
ESEADA "
190 LET K$=INKEY$
195 IF K$="0" THEN  GO TO 10000
200 IF CODE K$<49 OR CODE K$>52 THEN  BEEP .05,10: G
O TO 180
210 BEEP .1,40
220 PRINT AT 18,3;"
225 PLOT 0,18
230 DRAW 255,0
235 PLOT 0,19
240 DRAW 255,0
245 PLOT 0,5
250 DRAW 255,0
255 PLOT 0,4
260 DRAW 255,0
270 LET A$="INTRODUCE FUNCION f(x)"
280 FOR N=1 TO LEN A$
290 PRINT  INK 6;AT 20,N+3;A$(N)
300 BEEP .01,40
310 NEXT N
360 INPUT  LINE G$
365 CLS
370 GO SUB 400
380 GO TO (VAL K$+4)*100
400 REM [INV] DIBUJO COORDENADAS [TRUE]
405 FOR N=8 TO 21
410 PRINT  PAPER 1;AT N,0;"
                          "
415 NEXT N
420 PLOT 0,0: DRAW 255,0
425 DRAW 0,111
430 DRAW -255,0
435 DRAW 0,-111
440 PLOT 21,5
445 DRAW 0,101
450 PLOT 21,55
455 DRAW 200,0
460 FOR N=71 TO 221 STEP 50
465 PLOT N,57
470 DRAW 0,-4
475 NEXT N
480 RETURN
500 REM [INV] SENO [TRUE]
505 LET F$="SIN ("+G$+")"
510 LET N$="1"
515 GO TO 1100
600 REM [INV] COSENO [TRUE]
605 LET F$="COS ("+G$+")"
610 LET N$="1"
615 GO TO 1100
700 REM [INV] TANGENTE [TRUE]
705 LET F$="SIN ("+G$+")"
710 LET N$="COS ("+G$+")"
715 GO TO 1100
800 REM [INV] COTANGENTE [TRUE]
805 LET F$="COS ("+G$+")"
810 LET N$="SIN ("+G$+")"
815 GO TO 1100
1090 REM [INV] CALCULO FUNCION [TRUE]
1100 LET W$="("+F$+")"+"/"+"("++N$+")"
1102 GO SUB 2000
1105 LET A=0
1110 FOR X=0 TO 2*PI STEP PI/100
1120 LET A=A+1
1125 IF VAL N$=0 THEN  GO TO 1210
1130 LET Y(A)=VAL W$
1140 LET P(A)=55+INT (50*Y(A))
1155 IF P(A)>106 OR P(A)<5 THEN  GO TO 1190
1180 PLOT  OVER 1;A+20,P(A)
1190 PRINT AT 4,0;"Pix. X=";A+20;"   "
1191 PRINT AT 6,0;"Pix. Y=";P(A);"   "
1193 PRINT AT 4,14;"Grad.=";180*X/PI;"
1194 PRINT AT 6,15;"  Y=";Y(A);"                 "
1199 NEXT X
1200 GO TO 1240
1210 PLOT A+20,5
1220 DRAW 0,101
1230 GO TO 1190
1240 PRINT #0;"   PULSA UNA TECLA PARA MENU
1250 LET K$=INKEY$
1260 IF K$="" THEN  GO TO 1250
1270 RUN
2000 REM [INV]DISPL. DATOS[TRUE]
2010 PRINT AT 0,11; PAPER 2; INK 6;" FUNCION "
2020 PRINT
2030 PRINT TAB 1;"Y= ";W$
2040 RETURN
```

PROCESADORES DE TEXTO

INICIAMOS con este capítulo un ciclo dedicado al tratamiento de las aplicaciones comerciales más usuales en cualquier ordenador. Nada más propio en una obra de estas características que comenzar esta serie de artículos tratando un programa destinado precisamente a la transcripción escrita de documentos: los procesadores de textos.

El Spectrum es sin duda uno de los ordenadores que dispone en la actualidad de una mayor variedad de programas, no sólo versando sobre temáticas muy diferentes, sino también ofreciendo una gran variedad en aplicaciones con objetivos muy similares; sin embargo, dentro de los denominados procesadores de textos, este fenómeno no se produce.

Un magnífico programa: TASWORD TWO ha venido a satisfacer completamente las necesidades del usuario del Spectrum en el ámbito de los procesadores de textos, no dejando opción al nacimiento de ningún otro programa en este sentido, por ser prácticamente imposible aportar mayores prestaciones.

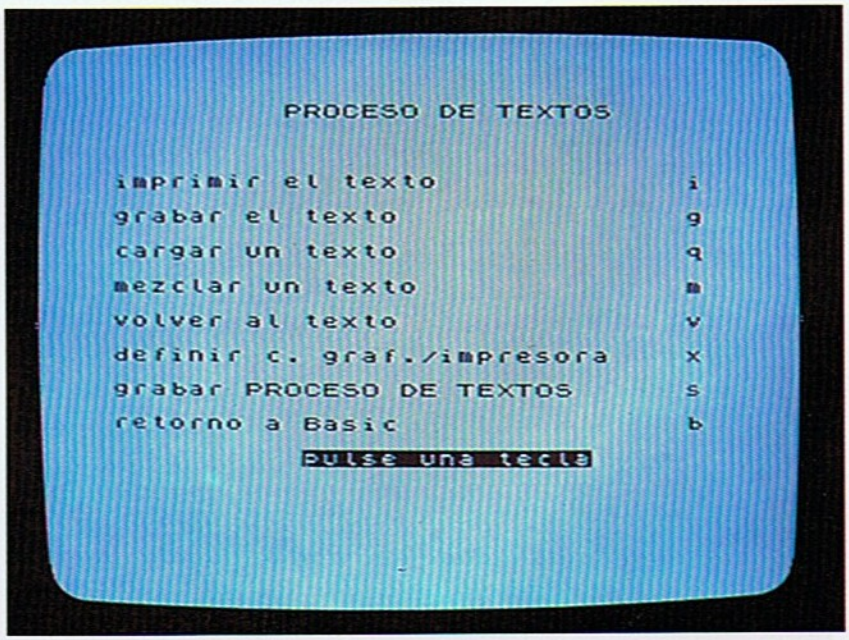


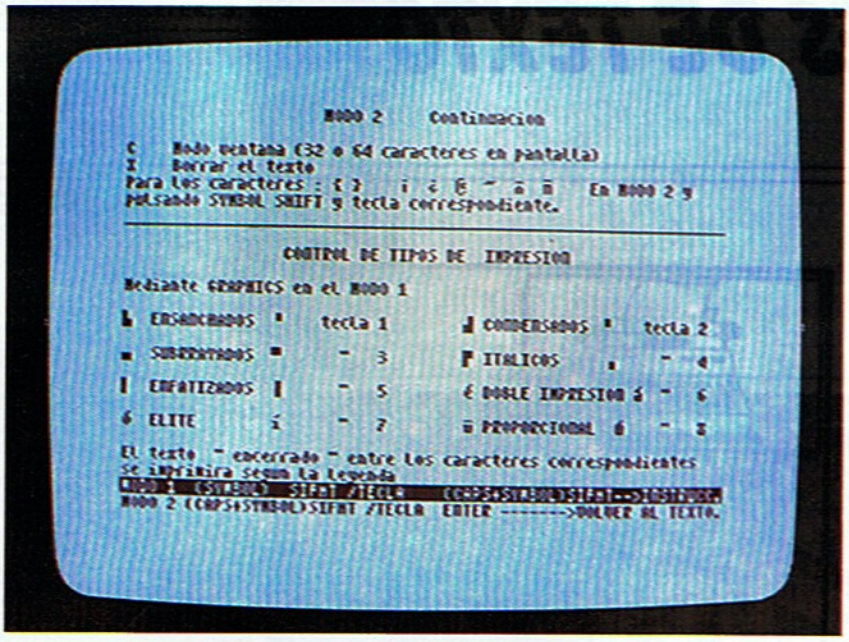
Un procesador de textos es cualquier paquete software que tenga por objetivo la entrada de textos para su posterior elaboración.

¿QUE ES UN PROCESADOR DE TEXTOS?

Seguramente que todos nosotros tenemos una idea, aunque sea aproximada, de las características, funciones y objetivos de un procesador de textos. Para asentar estos conocimientos básicos realizaremos una descripción de los aspectos más relevantes de una aplicación de esta índole. En general, un procesador de textos es cualquier paquete de *software* que tenga por objetivo la entrada de textos, ya sea directamente desde el teclado, o partiendo de un soporte magnético o de otro tipo, para su posterior elaboración siguiendo unas normas muy variadas, y salida del mencionado texto una vez procesado, usualmente en forma de escrito, aunque también a través de otros medios como puede ser el propio soporte magnético.

Desde el modo de edición podemos acceder a dos pantallas resumen de instrucciones.





El TASWORD TWO posibilita la edición de textos contínuos (sin word wrapping ni justificación), en bandera (con word wrapping pero sin justificación) y justificados (ambas caracteristicas).

Sin duda alguna, el punto más árduo de todo el proceso seguido es la elaboración de los datos introducidos para su posterior salida. Es en esta faceta en la que se diferencian por sus prestaciones los diversos programas de este tipo. Veamos entonces cuáles son las características más habituales de dichos programas en esta fase del proceso.

Esencialmente la pantalla de nuestro ordenador se convierte en una ventana de una inmensa e imaginaria hoja de papel sobre la cual podemos escribir, borrar, y modificar los textos escritos a nuestro antojo. Para ello se establecen determinadas secuencias de teclas que, en forma de comandos, ejecutan las distintas funciones de las cuales esté dotado el procesador en cuestión.

Así por ejemplo, podremos desplazar la ventana a lo largo del texto, delimitar el número de caracteres que se nos muestran a un mismo tiempo (desplazar la ventana no sólo a lo largo sino a lo ancho de la hoja), efectuar la justificación mediante la inclusión automática de espacios en las líneas de texto introducidas, fijación de tabuladores y márgenes de escritura, y algunas otras funciones un tanto más especializadas.

En este sentido, disponemos de la posibilidad de borrar zonas extensas de texto, insertar líneas en blanco entre zonas ya redactadas copiar bloques de líneas, desplazar párrafos de texto desde un punto a otro del mismo, buscar palabras, reemplazar de forma automática unas por otras, y un sin fin de aplicaciones entre las que se cuentan las propias de las características de salida del texto.

En este último sentido, podemos marcar el tipo de letra a emplear para la salida impresa del documento, el número de copias que de éste se requieren, el espaciado entre líneas a utilizar... O bien, en lo referente al almacenamiento sobre soporte magnético la grabación de copias de seguridad, el encadenamiento o fusión de diferentes textos, la reedición de los mismos o la destrucción de las copias desechadas, así como la combinación del proceso de textos con otros programas tales como bases de datos o ficheros particulares que nos permitan efectuar la edición de cartas personalizadas, circulares, etc...

Una vez terminada esta revisión sobre las características presentes habitualmente en las aplicaciones de proceso de textos, pasemos a estudiar concretamente las prestaciones que nos ofrece el destinado a nuestro ordenador.

Algunas versiones para el mercado español del TASWORD TWO incorporan no sólo caracteres propios de nuestro idioma, sino también la posibilidad de manejo del MICRODRIVE e impresoras semi-profesionales.

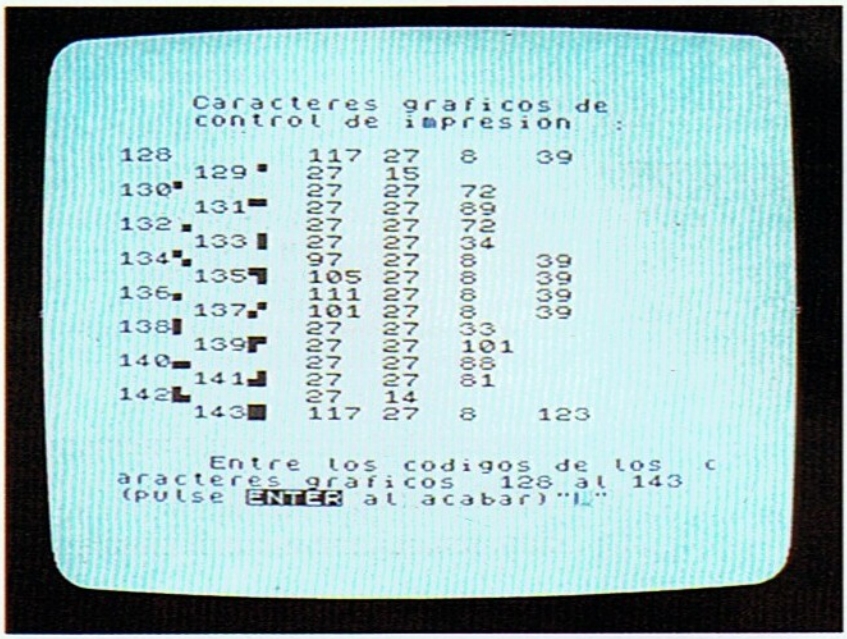
TASWORD TWO

En el año 1983 la firma TASMAN SOFTWARE lanza al mercado su programa TASWORD TWO con el apabullante subtítulo de *the word proces-*

sor (el procesador de textos). Este determinante calificativo no resulta en absoluto un abuso de confianza en su producto, y la reacción del mercado ante el mismo es buena prueba de ello.
De hecho, como se ha mencionado anteriormente, en el mercado nacional no existen más que versiones sobre este mismo programa que lo modifican en aspectos muy concretos y bastante superficiales, aunque no en su esencia; tales como adaptaciones de la Aplicación al empleo de MIDRODRIVE, etc... que más adelante repasaremos con mayor detenimiento.
En su forma original, TASWORD TWO es un programa compuesto por un soporte BASIC de reducido tamaño que lleva a cabo funciones auxiliares y un bloque de rutinas en código máquina conformando el núcleo principal del programa.
La aplicación está controlada desde el BASIC por un menú general a través del cual se tiene acceso a las siguientes opciones:

— IMPRESION.
— GRABACION.
— LECTURA.
— ENCADENAMIENTO.
— EDICION.
— DEFINICION DE CARACTERES Y CODIGOS DE IMPRESORA.
— COPIA DEL PROGRAMA.
— ACCESO AL BASIC.

El primer punto realiza la impresión del documento actualmente en memoria en cualquier impresora adaptada al programa. Inicialmente, éste puede trabajar con la ZX PRINTER, aunque mediante la opción de definición de códigos de impresora es posible controlar periféricos de tipo profesional a través de los correspondientes interfaces, como más adelante veremos.
Las tres siguientes opciones del menú son bien simples puesto que efectúan la grabación, carga o fusión en memoria de textos almacenados sobre soporte magnético (casete en la versión original). Como datos adicionales al realizar una grabación el programa indica la longitud de éste en líneas y el número de bytes de memoria empleados (este siempre coincide con el producto de 64×el número de líneas).
Asimismo, y como factor de seguridad, una vez completada la grabación se da opción a la verificación de dicho proceso. Por su parte, la lectura de un texto borra automáticamente cualquiera que existiera en memoria con anterioridad pasando a cargar el nuevo a partir de la primera línea, mientras que la fusión añade el leído inmediatamente a continuación del presente en memoria (al principio si no hubiera ninguno).
La definición de caracteres y códigos de impresora nos permite asignar a determinados caracteres gráficos accesibles por el teclado, coincidentes con los serigrafiados sobre las teclas numéricas, códigos para impresora que no constituyen caracteres representables sino la activación de determinadas funciones de la misma como son el salto automático de página, impresión en enfatizado, itálica, subrayado, etc...
Lógicamente, estas características sólo son posibles en impresoras de una cierta calidad, pero no en la ZX PRINTER. Precisamente pensando en



Durante el proceso de edición, se muestra constantemente en la zona inferior de la pantalla información sobre la posición del cursor y las características de la edición.

La ventana mostrada usualmente por el TASWORD es de 22 líneas de 64 caracteres sobre un máximo de 300 líneas disponibles.

¡!

La aplicación está controlada desde el BASIC por un menú general a través del cual se tiene acceso a las siguientes opciones: IMPRESION, GRABACION, LECTURA, ENCADENAMIENTO, EDICION, DEFINICION DE CARACTERES Y CODIGOS DE IMPRESORA, COPIA DEL PROGRAMA, ACCESO AL BASIC.

*

El texto completo está compuesto por un máximo de 300 líneas de 64 caracteres de ancho, a lo largo de los cuales nos podremos desplazar mediante el uso de los cursores produciendo los correspondientes *scrolling* para visualizar permanentemente en pantalla el punto y entorno donde nos encontramos.

*

La definición de caracteres y códigos de impresora nos permite asignar a determinados caracteres gráficos códigos para impresora que no constituyen caracteres representables sino la activación de determinadas funciones de la misma como son el salto automático de página, impresión en enfatizado, itálica, subrayado, etc...

este tipo de impresoras semi-profesionales esta misma opción del menú permite el control de los interfaces comerciales más habituales: Kempston e Indescomp.

Los dos últimos puntos del menú se explican por sí solos. La grabación del programa nos permite extraer copias de seguridad del mismo, completamente operativas pero que jamás deberemos emplear para vulnerar los derechos de autor del mismo. Por otra parte, la opción de retorno al BASIC nos permite el acceso a éste donde podemos incluir nuevas partes del programa con funciones de particular interés para nuestro uso.

EDICION DE TEXTOS

Intencionadamente, hemos pospuesto el comentario de esta opción por ser la más importante del programa al constituir propiamente dicho su fase de proceso de textos. Los anteriores puntos del menú eran gestionados desde el BASIC, lo cual nos puede permitir, dotados de cierto nivel de conocimientos, modificar las características de las mismas. Por el contrario, las rutinas de edición se encuentran íntegramente programadas en código máquina, siendo por tanto de más difícil acceso.

Al entrar en el modo de edición se presenta ante nosotros una ventana sobre el texto completo conformada por 22 líneas de 64 caracteres, estando reservadas la 23 y 24 de la pantalla, para información del Sistema tal como: línea y columna en que se encuentra el cursor, estado activo o pasivo de los modos de justificación automática, inserción, fijación de mayúsculas o edición de gráficos.

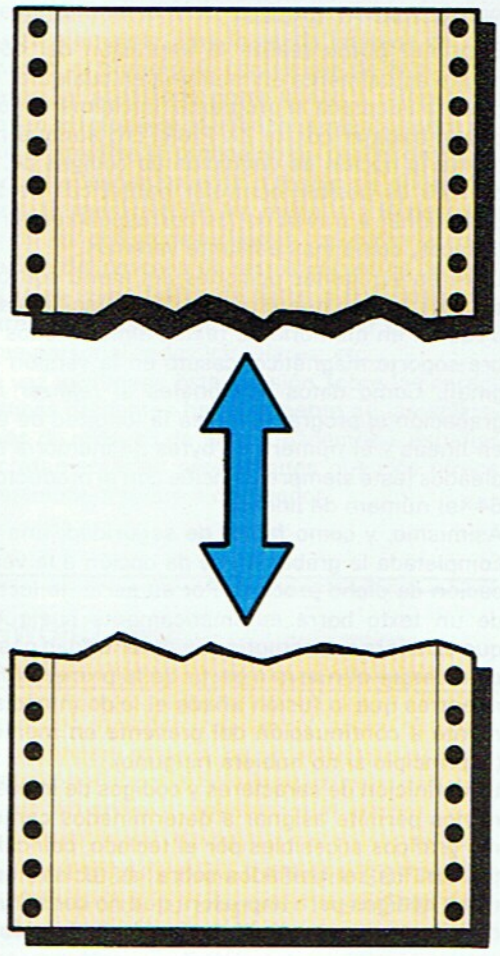
El texto completo está compuesto por un máximo de 300 líneas de 64 caracteres de ancho, a lo largo de los cuales nos podremos desplazar mediante el uso de los cursores produciendo los correspondientes *scrolling* para visualizar permanentemente en pantalla el punto y entorno donde nos encontramos. Dicha posición viene siempre marcada por un cursor intermitente representado por el carácter gráfico de la tecla 3.

Aunque la Aplicación se acompaña de un manual de considerable extensión, mediante la pulsación de **EDIT** en combinación con **EXTENDED MODE** tenemos acceso a dos pantallas resumen de las funciones de edición, que como veremos a continuación no son pocas.

Como ya hemos mencionado, las flechas nos permiten el desplazamiento por el texto carácter a carácter, **TRUE VIDEO** el retroceso palabra por palabra, e **INVERSE VIDEO** el avance del mismo tipo. **DELETE** efectúa el borrado del carácter anterior, con desplazamiento del resto de la línea una posición a la izquierda, **CAPS LOCK** la conexión/desconexión del modo mayúsculas y **GRAPHICS** función similar para el modo gráfico, mientras que **ENTER** efectúa un retorno de carro (paso al comienzo de la línea siguiente) y la pulsación combinada de las teclas **CAPS SHIFT** y **SYMBOL SHIFT** al modo **EXTENDED**. Para desplazamientos a zonas más lejanas de la ventana de texto se habilitan los siguientes controles: **TO** *(scroll adelante una línea)* **THEN** *(scroll* atrás una línea), **EXTENDED F** *(scroll* adelante una página, 22 líneas), **EXTENDED G** *(scroll* atrás una página), **AT** desplazamiento a la primera línea del texto y **OR** a la última.

Cada vez que se alcanza el final de una línea ésta es automáticamente justificada a la derecha con inserción de espacios adicionales entre palabras

Entre las funciones que desarrolla el TASWORD TWO nos encontramos con la inserción de líneas en blanco entre zonas ya redactadas.



de derecha a izquierda, hasta completar la longitud máxima de la misma, antes de posicionar el cursor al comienzo de la siguiente línea. Esta posibilidad puede ser deshabilitada por medio de **EXTENDED J,** quedando entonces los textos dispuestos en bandera por efecto del *word wrapping* (salto automático a la línea siguiente de la última palabra que no cabe completa en la anterior). Esta última función también puede ser desactivada **(EXTENDED W),** con lo cual los textos serán escritos en modo continuo sin tenerse en cuenta para nada la longitud de las líneas.

Para la inserción de caracteres o palabras entre líneas ya justificadas utilizamos **AND,** así como para insertar párrafos podemos hacer uso del modo de inserción automático pulsando **EXTENDED I.** La función inversa, es decir, la eliminación de líneas completas, se consigue mediante el empleo de **NOT.**

En cuanto al tratamiento por línea, éstas pueden justificarse **(EXTENDED J)** y desjustificarse **(EXTENDED H)** individualmente, así como centrarse **(<>),** desplazarse hacia la izquierda carácter a carácter (<=) o hacia la derecha (=>). También es posible justificar todo un párrafo (desde la línea en que se encuentra el cursor hasta el próximo punto y aparte) mediante **STEP.**

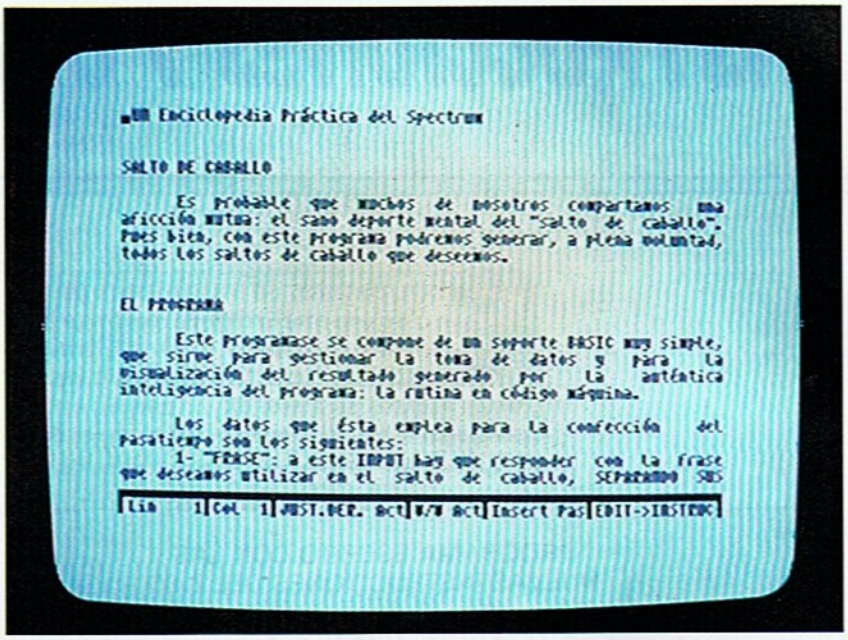
En cuanto al formato de presentación la fijación del margen izquierdo se consigue a través de **EXTENDED A,** la del derecho con **EXTENDED D** y la restauración de ambos con **EXTENDED S.** Asimismo, **EXTENDED C** visualizará una ventana de texto de 22 líneas por 32 caracteres de tamaño normal, con lo cual la escritura de un texto irá produciendo automáticamente el desplazamiento de la misma para proporcionar una visión de mayor entorno posible.

Por último, antes de entrar en las características más especiales del procesador, cabe mencionar que el borrado del texto en memoria se consigue por medio de **EXTENDED X** (con un mensaje permiso de validación) y la salida al menú de opciones mediante **STOP.**

Las versiones nacionales implementan, generalmente a través de los caracteres gráficos, signos propios del alfabeto español como son la apertura de interrogación y admiración, letra eñe mayúscula y minúscula, vocales acentuadas y u con diéresis.

Seguramente las características más espectaculares del procesador dentro del modo edición son la posibilidad de desplazar y copiar bloques de texto y la de la búsqueda y búsqueda con reemplazo de una palabra por otra o una frase.

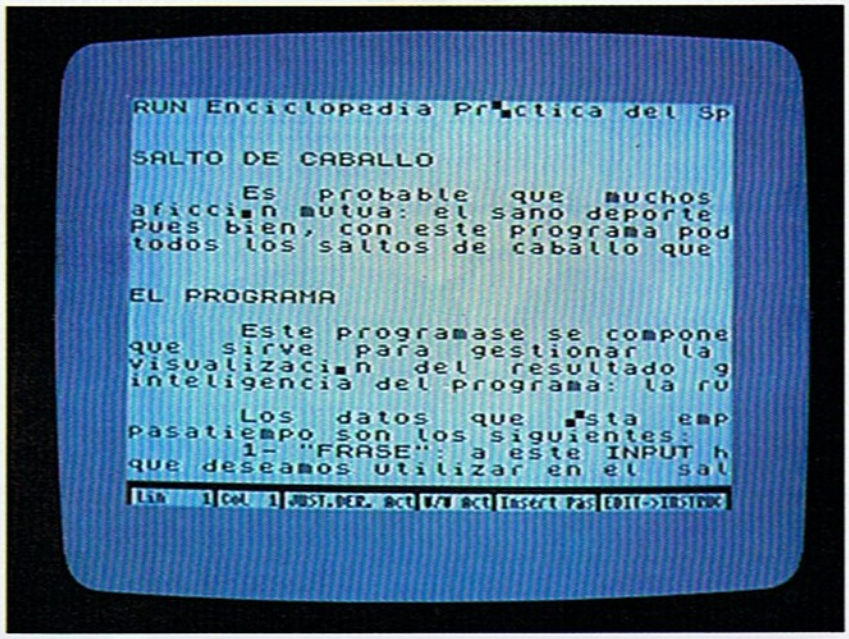
Para el tratamiento de bloques se ha de fijar previamente el límite superior **(EXTENDED B)** e inferior del mismo **(EXTENDED V)** y situar el cursor en el punto al cual se desee transferir el texto, pulsando **EXTENDED M** para el desplazamiento del bloque y **EXTENDED N** para la reproducción del mismo.



La pulsación de EXTENDED C *nos permite la entrada o salida del modo «ventana de 32 caracteres».*

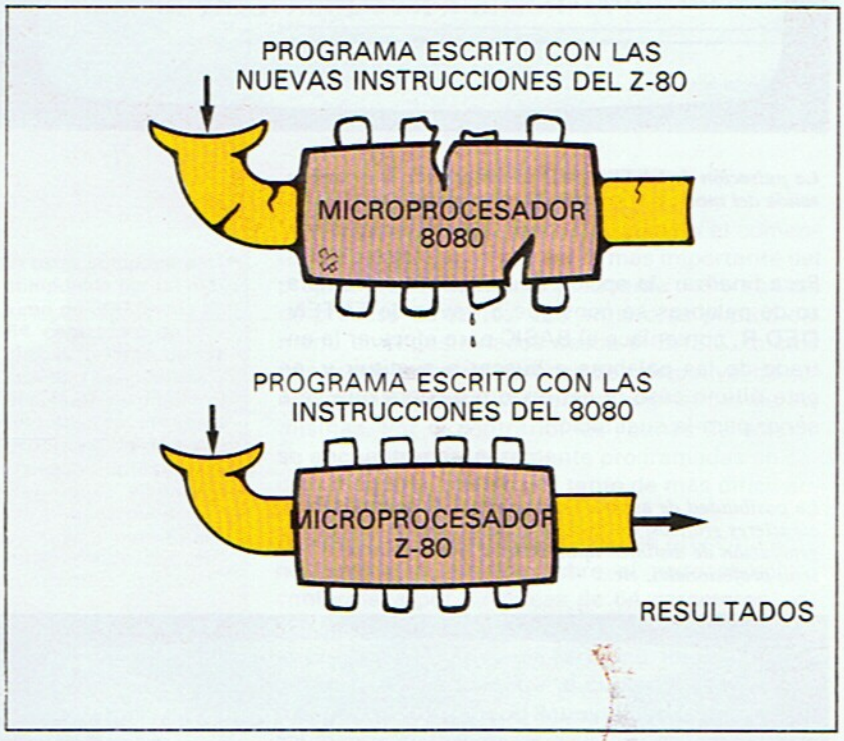
Para finalizar, la opción de búsqueda o reemplazo de palabras se consigue a través de **EXTENDED R,** con enlace al BASIC para efectuar la entrada de las palabras a buscar o sustituir y, en este último caso la del texto o palabra que va a servir para la sustitución.

La posibilidad de asignación de códigos de control a los caracteres gráficos, da entrada a características como la generación de distintos tipos de letra en impresoras semi-profesionales, etc...



INSTRUCCIONES Y OPERACIONES

EL conjunto de instrucciones del Z80 es una ampliación del presente en el microprocesador 8080 de la firma INTEL. Por ello, la mayoría de los programas escritos para este último pueden ser ejecutados en el desarrollado por los técnicos de ZILOG en el año 1976.



Es más, la compatibilidad podría hacerse efectiva a la inversa, siempre y cuando no incluyéramos en nuestros programas las nuevas instrucciones implementadas. No es este, en principio, nuestro objetivo, pues son precisamente los nuevos grupos los que confieren una potencia fuera de común a este singular microprocesador de 8 bits.

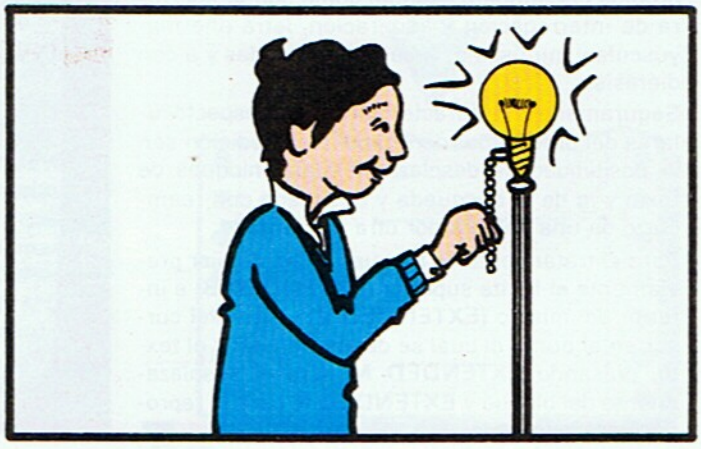
En el capítulo anterior tratamos muchos nuevos conceptos y ya es hora de formalizar claramente su significado. Continuemos, por tanto, con algunas definiciones útiles.

A nadie se le oculta que cuando hablamos de programación, cualquier instrucción está formada por un grupo de caracteres que definen una operación. Un carácter es cualquier símbolo de los que nosotros consideramos elementales, como una letra del alfabeto, un número del 0 al 9, o los especiales: dólar ($), asterisco (*), entre otros.

Una operación es la acción específica que el ordenador ejecuta cuando una instrucción se lo exije. Por tanto, cualquier instrucción en C/M estará compuesta, entre otras cosas que en seguida

Los programas escritos para el 8080 pueden ejecutarse en el Z80, pero el proceso inverso no siempre es posible.

Instrucción: definición de la operación. Operación: acción específica a ejecutar.



comentaremos, por uno o varios códigos de operación, los cuales especifican al Z80 lo que debe hacer en todo momento.

Formato de las instrucciones del Z80.

LOS FORMATOS

Las tablas que nos acompañan en estas páginas contienen el conjunto completo de instrucciones

1. INSTRUCCIONES DE UN BYTE:

CODIGO DE OPERACION	DIR

2. INSTRUCCIONES DE DOS BYTES:

CODIGO DE OPERACIÓN	DIR	CODIGO DE OPERACION	CODIGO DE OPERACION	CODIGO DE OPERACION	CODIGO DE OPERACION	CODIGO DE OPERACION	CODIGO DE OPERACION
CODIGO DE OPERACION	DIR+1	DATO		CODIGO DE DISPOSITIVO		DESPLAZAMIENTO	

3. INSTRUCCIONES DE TRES BYTES:

C.O.	DIR	C.O.	C.O.	
DATO	DIR+1	DIRECC. LO	C.O.	
DATO	DIR+2	DIRECC. HI	DESPLAZAMIENTO	

4. INSTRUCCIONES DE CUATRO BYTES:

C.O.	C.O.	C.O.	C.O.
C.O.	C.O.	C.O.	C.O.
DATO	DESPLAZAMIENTO	DIRECC. LO	DESPLAZAMIENTO
DATO	C.O.	DIRECC. HI	DATO

que es posible implementar manejando el Z80, ordenadas según su función. Son 158 tipos diferentes (696 de diferente código de operación) y no debemos preocuparnos si encontramos algunos símbolos o datos que todavía no hayamos estudiado. Todo llegará en su momento.
En la figura se muestran los formatos de las instrucciones, atendiendo al número de octetos necesarios para su correcta construcción. Como observamos pueden estar compuestas por uno, dos, tres o cuatro bytes, extremo que podemos confirmar ojeando las tablas principales. Pero, ¿cómo es capaz de reconocer el microprocesador su longitud?
Si nos fijamos con detalle, el primer byte de todas las instrucciones es siempre un código de operación, sea cual sea el tamaño de ésta. El proceso seguido por la CPU de nuestro Spectrum es el siguiente:
— El registro PC le indica la dirección de memoria donde encontrar la siguiente instrucción. Allí lee un código de operacion, y en función de éste, reconoce el número de bytes de la instrucción, cantidad que es añadida al contador de programa PC. Finalmente, la instrucción es procesada y el Z80 busca un nuevo código de operación en la dirección señalada por el registro PC.
Algunas instrucciones, las cuales provocan saltos o bifurcaciones en el proceso secuencial de un programa (algo parecido a lo que ocurre con las instrucciones **GO TO** o **GOSUB** del BASIC), pueden modificar bruscamente el contenido del registro PC, pero siempre quedará almacenado en éste la dirección de memoria en donde la CPU debe buscar la siguiente instrucción a ejecutar.
Los siguientes octetos al primer código de operación si los hubiere pueden contener información de diversa índole, que a continuación detallamos:
— Otro código de operación de 8 bits, el cual indica a la CPU el tipo de instrucción de que se trata (transferencia de datos, entrada/salida, lógica, manejo del STACK, etc.).
— Un dato de 8 ó 16 bits, codificado en binario, ASCII, binario codificado en decimal (BCD), etc.
— Un código de dispositivo de 8 bits. Teóricamente el Z80 es capaz de direccionar hasta 2↑8=256 periféricos, pero esta cuestión está íntimamente relacionada con el diseño de los circuitos exteriores al microprocesador que en su momento trazaron sus fabricantes, quedando habilitadas sólo algunas líneas, las cuales soportan por ejemplo, el casete, el teclado o la impresora.
— Un byte de desplazamiento, es decir, un número en complemento a dos con signo, empleado en las instrucciones que manejan direccionamiento indexado (pronto aprenderemos a calcular el complemento a dos de un número).
— Una dirección determinada por dos bytes, donde el microprocesador siempre interpretará como byte de menor peso, LO (abreviatura de *LOw,* bajo), el presente en primer lugar, siendo el segundo el más significativo, HI (abreviatura de *HIgh,* alto).
Para todos ha de estar claro que el número en cuestión es interpretado por el Z80 de la siguiente manera:

VALOR TOTAL = VALOR DEL BYTE LO + 256 × VALOR DEL BYTE HI

TRANSFERENCIA

ORIGEN

DESTINO (DE)		INSTRUCCION
(HL)		
ED A0	LDI	
ED B0	LDIR	
ED A8	LDD	
ED B8	LDDR	

BUSQUEDA

DIRECCION DE BUSQUEDA	
(HL)	
ED A1	CPI
ED B1	CPIR
ED A9	CPD
ED B9	CPDR

ENTRADA/SALIDA

INA,(n)	DB n
OUT(n),A	D3 n
INI	ED A2
INIR	ED B2
IND	ED AA
INDR	ED BA
OUTI	ED A3
OTIR	ED B3
OUTD	ED AB
OTDR	ED BB

INTERCAMBIO (EX y EXX)

	AF'	BC' DE' HL'	HL	IX	
AF	08				
BC DE HL		D9			
DE			EB		
(SP)			E3	DD E3	FD E3

ENTRADA/SALIDA

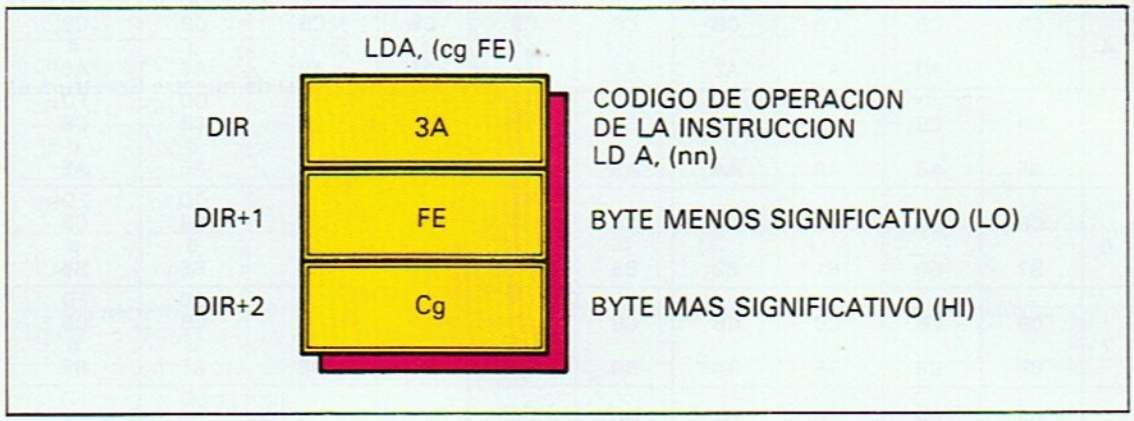
	A	B	C	D	E	H	L
IN	ED 78	ED 40	ED 48	ED 50	ED 58	ED 60	ED 68
OUT	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69

ARITMETICO DE 16 BITS

		BC	DE	HL	SP	IX	IY
ADD	HL	09	19	29	39		
	IX	DD 09	DD 19		DD 39	DD 29	
	IY	FD 09	FD 19		FD 39		FD 29
ADC	HL	ED 4A	ED 5A	ED 6A	ED 7A		
SBC	HL	ED 42	ED 52	ED 62	ED 72		
INC		03	13	23	33	DD 23	FD 23
DEC		0B	1B	2B	3B	DD 2B	FD 2B

Por ejemplo, consideremos la instrucción **LD A,(C9FE),** la cual «carga» el acumulador con el valor almacenado en la dirección C9FE H. En la figura se muestra el formato en que el ordenador debe encontrarla para su correcta interpretación. En el próximo capítulo encontraremos las indicaciones necesarias para obtener el máximo rendimiento de las tablas y realizaremos un recorrido a través de los diferentes grupos que las componen.

Cuando la CPU interpreta una dirección supone que ha encontrado los bytes que la definen en el orden.



MANIPULACION DE BITS

BIT	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E

El conjunto de instrucciones del Z80 es una ampliación del presente en el microprocesador 8080 de la firma INTEL.

El primer byte de todas las instrucciones es siempre un código de operación.

RES	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET	0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

RETORNO

		CONDICION								
		INCOND.	ARRASTRE	NO ARRASTRE	CERO	NO CERO	PARIDAD PAR	PARIDAD IMPAR	SIGNO NEG.	SIGNO POS.
RET	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0
RETI	(SP) (SP+1)	ED 4D								
RETN	(SP) (SP+1)	ED 45								

SALTOS Y LLAMADAS

		CONDICION									
		INCOND.	ARRASTRE	NO ARRASTRE	CERO	NO CERO	PARIDAD PAR	PARIDAD IMPAR	SIGNO NEG.	SIGNO POS.	B≠0
JP	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	IA n n	I2 n n	
JR	PC+e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JP	(HL)	E9									
JP	(IX)	DD E9									
JP	(IY)	FD E9									
CALL	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DJNZ	PC+e										10 e-2

DIRECCION DE LLAMADA	00 h	08 h	10 h	18 h	20 h	28 h	30 h	38 h
INSTRUCCION	RST 0	RST 8	RST 10	RST 18	RST 20	RST 28	RST 30	RST 38
CODIGO DE OPERACION	C7	CF	D7	DF	E7	EF	F7	FF

CARGA DE 16 BITS (LD)

		ORIGEN									POP
		AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
DESTINO	AF										F1
	BC								01 n n	ED 4B n n	C1
	DE								11 n n	ED 5B n n	D1
	HL								21 n n	2A n n	E1
	SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
	IX								DD 21 n n	DD 2A n n	DD E1
	IY								FD 21 n n	FD 2A n n	FD E1
	(n)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH	(SP)	F5	C5	D5	E5		DD E5	FD E5			

¡!

Los nuevos grupos de instrucciones implementadas en el Z80 son los que dotan de una potencia fuera de lo común a este microprocesador.

*

El Z80 es un microprocesador de 8 bits.

LOGICO 8 BITS

	ORIGEN										
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
AND	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
XOR	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
OF	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
CP	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n

CARGA DE 8 BITS (LD)

DESTINO	ORIGEN															
	I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+0)	(nn)	n
A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
(HL)			77	70	71	72	73	74	75							36 n
(BC)			02													
(DE)			12													
(IX+d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
(IY+d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
(nn)			32 n n													
I			ED 47													
R			ED 4F													

ROTACION Y DESPLAZAMIENTO

	ORIGEN Y DESTINO									
	A	B	C	D	E	H	L	(HL)	(IX+d)	(YI+d)
RLC	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
RRC	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
RL	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
RR	CB IF	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
SLA	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
SRA	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
SRL	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d	FD CB d
RLD								ED 6F		
RRD								ED 67		
RLCA	07									
RRCA	0F									
RLA	17									
RRA	IF									

ARITMETICO 8 BITS

	ORIGEN										
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
ADD	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADC	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUB	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SBC	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
INC	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DEC	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

ARITMETICO Y DE CONTROL DE LA C.P.U.

DAA	27
CPL	2F
NEG	ED 44
CCF	3F
SCF	37
NOP	00
HALT	76
DI	F3
EI	FB
IM0	ED 46
IM1	ED 56
IM2	ED 5E

IMPRENTA

¡!

Si somos observadores nos daremos cuenta de que las letras minúsculas empleadas en los juegos ITALICA y ROTULACION son iguales, por lo que podremos aprovechar con habilidad el esfuerzo de introducción si deseamos disponer de ambos.

*

En el caso de los juegos de caracteres, al estar íntegramente constituidos por datos, la reubicabilidad es completa, es decir, no importa el lugar de memoria en que los grabemos, podrán ser situados en cualquier punto de la memoria.

Como sabemos, el juego de caracteres que utiliza el Spectrum (mayúsculas, minúsculas y caracteres especiales) están almacenados en ROM en grupos de 8 bytes, en donde cada uno de estos octetos equivale a una línea del *pattern* (forma del carácter).

Entre las muchas Variables del Sistema que nuestro Spectrum posee, existe una que le indica al microprocesador la dirección de comienzo del generador de caracteres. **CHARS** es nombre simbólico que Sir Clive Sinclair le ha dado a esta variable, y que no tiene nada que ver con las variables que nosotros definimos en BASIC.

Inicialmente, el puntero de la variable **CHARS** señala a una dirección de la ROM, pero en contra de lo que pudiéramos pensar, dicha dirección no indica el comienzo del generador de caracteres, sino 256 bytes antes. Esto es debido a que los primeros 32 caracteres no son representables (8 bytes por 32 caracteres=256 bytes).

Al encontrarse en zona RAM el área de Variables del Sistema, podemos «engañar» a nuestro Spectrum modificando la información que contiene la variable **CHARS**, e introduciendo en la dirección adecuada nuestro propio juego de caracteres.

Sin duda, el problema más grave es disponer de un juego de caracteres definido por el cual poder sustituir el propio de Sinclair. Recrearse la forma de tantos caracteres es sin duda un árduo trabajo, aunque para animarnos incluímos en este artículo nada más y nada menos que cuatro juegos diferentes, de los cuales tenemos una muestra por impresora, para que podamos realizar nuestra elección con conocimiento de causa.

En cuanto a la introducción de los datos correspondientes a cada juego, podemos utilizar cualquier cargador de código máquina, como por ejemplo el que reproducimos en estas páginas y cuyo funcionamiento es de extremada sencillez. Una vez introducido (podemos convervarlo para alguna otra ocasión en que necesitemos una rutina similar) y ejecutado, aparecerá en la pantalla la petición de DIRECCION DE INICIO. Esta supone la dirección de memoria a partir de la cual se comenzarán a introducir los datos.

En nuestro caso, como se puede apreciar en los listados de los juegos de caracteres, hemos seleccionado la dirección 30720, por ser múltiplo de 256 y válida para los modelos de 16K y 48K, aunque al tratarse de datos esta dirección carece de importancia decisiva, siempre y cuando no la escojamos tan baja que altere el propio programa BASIC, o alguna otra zona de memoria de importancia.

Acto seguido se efectúa la toma de datos byte a byte, que deberán ser introducidos, tal como aparecen en el listado, de izquierda a derecha y de arriba a abajo, siendo el número de la columna izquierda un mero indicativo de la dirección a partir de la cual se efectúa la carga, para servirnos de guía entre el maremagnum de cifras (algo así como las piedras de Pulgarcito).

Como característica adicional de la rutina, si entramos el código **E**, se tratará como una detección de error, y el programa retrocederá un paso en la introducción de los datos. Asimismo, el código **F** se ha habilitado para la notificación de fin de trabajo, lo cual dará acceso a una pregunta sobre el nombre con el cual deseamos realizar la grabación.

De esta sencilla manera podremos hacernos con los juegos de caracteres que acompañan este artículo.

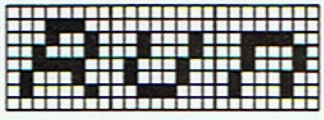
A continuación exponemos tres pequeños programas con los cuales, una vez definido el nuevo juego de caracteres, podremos:

1. Calcular el nuevo contenido de la variable **CHARS** en función de la dirección a partir de la cual queremos almacenar el nuevo juego de caracteres. (CALCULADOR.)

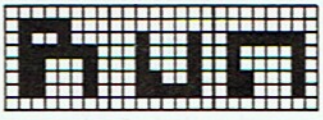
2. Cargar los valores que definen los *patterns*. (GENERADOR.)

3. Reubicar el juego de caracteres. (REUBICADOR.)

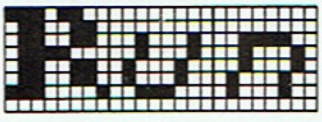
MARGARITA



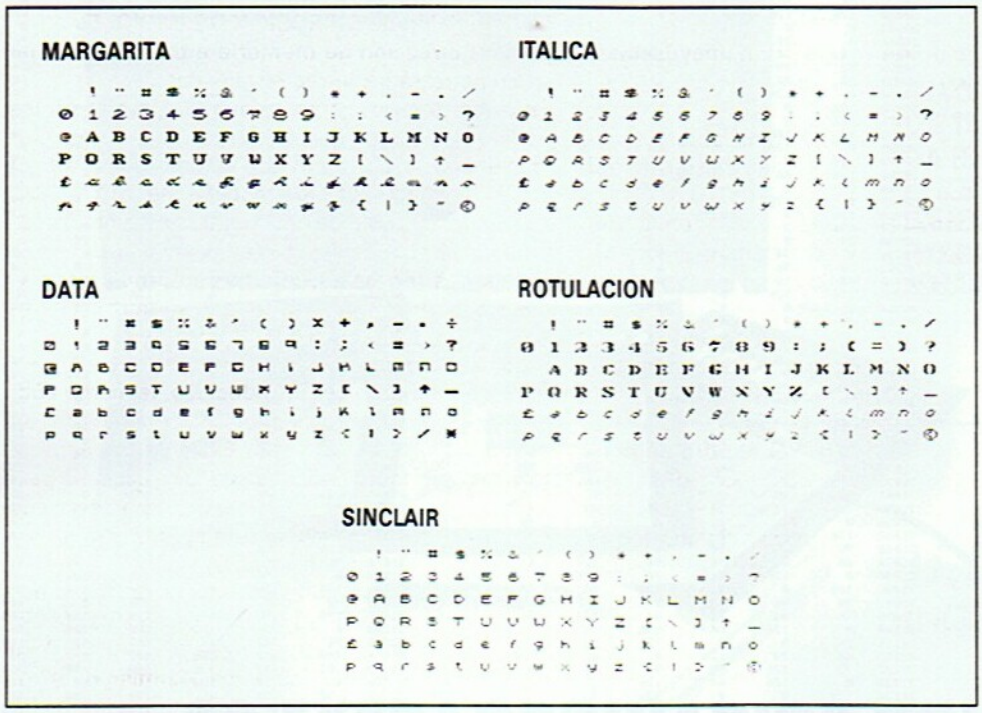
ITALICA



DATA



ROTULACION



PROGRAMA CALCULADOR

```
  1 REM ****** RUN ******
 10 PRINT AT 20,0;"INTRO. DIR. DE ALMACE
NAMIENTO"
 20 INPUT "DIRECCION= ": DIR
 30 LET CHARS=DIR-256
 40 LET B=INT (CHARS/256)
 50 LET A=CHARS-256*B
 60 PRINT "DIR. ALMACEN.--";DIR
 70 PRINT "DIREC. CHAR.--";CHAR
 80 PRINT,"POKE ";23606;",";A
 90 PRINT,"POKE ";23607;",";B
100 PRINT
110 POKE 23692,255
120  GOTO 10
```

PROGRAMA REUBICADOR

```
  1 REM ******* RUN *******
 10 INPUT "DIREC. ORIGEN ";D1
 20 INPUT "DIREC. NUEVA ";D2
 30 FOR N=0 TO 1023
 40 POKE (D2+N), PEEK (D1+N)
 50 NEXT N
 60 STOP
```

PROGRAMA GENERADOR

```
  1 REM ******* RUN ********
 10 INPUT "DIREC. COMIENZO ";D
 20 FOR N=0 TO 1023
 30 PRINT AT 10,3;"DIRECCION ";N+D
 40 INPUT "CONTENIDO ";C
 50 POKE (D+N),C
 60 NEXT N
 70 STOP
```

PROGRAMA CARGADOR

```
  10 REM FERNANDO LOPEZ MARTINEZ $ CARGADOR DE CODIGO
MAQUINA $ CARACTERES
  20 LET B$=CHR$ 8+CHR$ 8+CHR$ 8+CHR$ 8+"   "+CHR$ 8+
CHR$ 8+CHR$ 8: DEF FN P$()=("  "+A$)(LEN A$ TO )
  30 POKE 23658,8: BORDER 1: PAPER 1: INK 7: CLS
  40 INPUT "DIRECCION DE INICIO ?";A: LET P=A
  50 PRINT A;TAB 7;
  60 FOR I=1 TO 6
  70 INPUT "BYTE ?"; LINE A$
  80 IF A$="F" THEN  GO TO 160
  90 IF A$="E" THEN  GO TO 230
 100 IF VAL A$<0 OR VAL A$>255 THEN  GO TO 40
 110 POKE A,VAL A$
 120 LET A=A+1
 130 PRINT FN P$();" ";
 140 NEXT I: PRINT
 150 GO TO 50
 160 INPUT "NOMBRE DEL FICHERO ?";A$
 170 IF A$="" THEN  GO TO 30
 180 IF LEN A$>10 THEN  GO TO 160
 190 SAVE A$CODE P,A-P+1
 200 PRINT #0;"     PREPARE LA VERIFICACION"
 210 VERIFY A$CODE P,A-P+1
 220 RUN
 230 IF I=1 THEN  PRINT AT 23-PEEK 23689,31;: LET I=7
 240 PRINT B$;
 250 LET A=A-1: LET I=I-1
 260 GO TO 70
```

MARGARITA

30720	0	0	0	0	0	0
30726	0	0	24	24	24	24
30732	8	0	8	0	0	36
30738	36	0	0	0	0	0
30744	0	36	126	36	36	126
30750	36	0	8	126	200	126
30756	17	254	16	0	0	98
30762	100	8	16	38	70	0
30768	16	40	68	40	90	132
30774	122	0	0	8	16	0
30780	0	0	0	0	4	8
30786	8	8	8	8	4	0
30792	32	16	16	16	16	16
30798	32	0	0	0	20	8
30804	62	8	20	0	0	0
30810	8	8	62	8	8	0
30816	0	0	0	0	0	8
30822	8	16	0	0	0	0
30828	62	0	0	0	0	0
30834	0	0	0	24	24	0
30840	1	2	4	8	16	32
30846	64	0	60	70	203	219
30852	211	98	60	0	8	24
30858	40	8	8	8	62	0
30864	56	68	6	20	112	193
30870	254	0	124	131	3	12
30876	131	131	124	0	0	24
30882	56	72	255	0	0	0
30888	126	192	252	2	3	130
30894	124	0	62	64	252	194
30900	131	66	60	0	34	127
30906	129	18	12	18	16	0
30912	60	66	195	126	129	129
30918	126	0	60	66	193	67
30924	63	2	124	0	0	0
30930	16	0	0	16	0	0
30936	0	0	16	0	0	16
30942	16	32	0	0	4	8
30948	16	8	4	0	0	0
30954	0	62	0	62	0	0
30960	0	0	16	8	4	8
30966	16	0	60	102	67	6
30972	12	0	8	0	0	60
30978	74	86	94	64	60	0
30984	24	24	44	44	126	70
30990	239	0	252	102	102	124
30996	102	102	252	0	60	98
31002	96	96	96	98	60	0
31008	252	102	102	102	102	102
31014	252	0	254	98	104	120
31020	106	98	254	0	254	98
31026	104	120	104	96	240	0
31032	60	102	96	110	102	102
31038	60	0	255	102	102	126
31044	102	102	255	0	60	24
31050	24	24	24	24	60	0
31056	126	12	12	12	12	140
31062	120	0	242	100	104	120
31068	108	102	242	0	240	96
31074	96	96	98	98	254	0
31080	199	110	118	86	86	70
31086	239	0	199	98	114	90
31092	78	70	226	0	60	102
31098	102	102	102	102	60	0
31104	252	102	102	124	96	96
31110	240	0	124	198	198	198
31116	198	214	124	0	252	102
31122	102	124	108	102	247	0
31128	60	98	112	60	14	102
31134	60	0	255	153	153	24
31140	24	24	60	0	239	70
31146	70	70	70	70	60	0
31152	111	70	70	44	44	44
31158	16	0	115	49	49	53
31164	53	63	18	0	119	38
31170	52	24	28	38	119	0
31176	115	98	52	24	24	24
31182	60	0	127	3	6	12
31188	24	48	127	0	24	16
31194	16	16	16	16	24	0
31200	128	64	32	16	8	4
31206	2	0	48	16	16	16
31212	16	16	48	0	0	16
31218	56	84	16	16	16	0
31224	0	0	0	0	0	0
31230	0	255	0	20	50	120
31236	48	50	124	0	0	0
31242	29	34	100	191	0	0
31248	14	18	36	58	98	189
31254	0	0	0	0	28	36
31260	96	191	0	0	1	2
31266	60	72	200	119	0	0
31272	0	0	28	36	120	159
31278	0	0	15	25	18	60
31284	248	87	144	224	0	0
31290	31	50	212	63	72	112
31296	7	9	18	60	220	71
31302	0	0	4	0	8	16
31308	48	223	0	0	2	0
31314	12	56	216	127	144	240
31320	14	18	28	52	124	167
31326	0	0	6	9	18	20
31332	56	223	0	0	0	0
31338	84	170	170	87	0	0
31344	0	0	44	90	182	39
31350	0	0	0	0	28	38
31356	101	184	0	0	0	0
31362	20	58	246	103	64	192
31368	0	0	30	38	253	8
31374	24	48	0	32	56	72
31380	152	31	0	0	0	4
31386	12	22	58	221	0	0
31392	6	12	31	56	88	143
31398	0	0	0	0	38	108
31404	236	55	0	0	0	0
31410	34	87	180	56	0	0
31416	0	0	187	218	90	108
31422	0	0	0	0	18	45
31428	76	179	0	0	0	0
31434	9	58	220	63	72	112
31440	0	60	40	92	159	36
31446	40	48	14	8	8	56
31452	8	8	14	0	0	8
31458	8	8	8	8	8	0
31464	112	16	16	28	16	16
31470	112	0	0	20	40	0
31476	0	0	0	0	60	66
31482	153	161	161	153	66	60

ITALICA

30720	0	0	0	0	0	0
30726	0	0	24	24	24	24
30732	8	0	8	0	0	36
30738	36	0	0	0	0	0
30744	0	36	126	36	36	126
30750	36	0	8	126	200	126
30756	17	254	16	0	0	98
30762	100	8	16	38	70	0
30768	16	40	68	40	90	132
30774	122	0	0	8	16	0
30780	0	0	0	0	4	8
30786	8	8	8	8	4	0
30792	32	16	16	16	16	16
30798	32	0	0	0	20	8
30804	62	8	20	0	0	0
30810	8	8	62	8	8	0
30816	0	0	0	0	0	8
30822	8	16	0	0	0	0
30828	62	0	0	0	0	0
30834	0	0	0	24	24	0
30840	1	2	4	8	16	32
30846	64	0	0	60	70	74
30852	148	164	120	0	0	48
30858	80	16	32	32	248	0
30864	0	28	34	4	56	64
30870	124	0	0	30	4	24
30876	4	72	56	0	0	6
30882	10	20	36	126	8	0
30888	0	30	16	60	2	68
30894	56	0	0	30	32	60
30900	66	68	56	0	0	30
30906	2	4	8	16	32	0
30912	0	28	36	56	68	68
30918	56	0	0	28	34	34
30924	28	4	56	0	0	0
30930	16	0	0	16	0	0
30936	0	0	16	0	0	16
30942	16	32	0	0	4	8
30948	16	8	4	0	0	0
30954	0	62	0	62	0	0
30960	0	0	16	8	4	8
30966	16	0	60	102	67	6
30972	12	0	8	0	0	60
30978	74	86	94	64	60	0
30984	0	12	18	34	62	66
30990	66	0	0	28	18	60
30996	34	66	124	0	0	28
31002	34	32	64	68	56	0
31008	0	24	20	34	34	68
31014	120	0	0	30	16	60
31020	32	64	120	0	0	30
31026	16	60	32	64	64	0
31032	0	28	34	32	76	68
31038	56	0	0	18	18	60
31044	36	72	72	0	0	62
31050	8	16	16	32	248	0
31056	0	2	2	4	68	72
31062	56	0	0	18	20	56
31068	40	68	66	0	0	16
31074	16	32	32	64	124	0
31080	0	34	54	42	66	68
31086	68	0	0	18	26	42
31092	44	68	68	0	0	28
31098	34	34	68	68	56	0
31104	0	28	18	34	60	64
31110	64	0	0	60	66	66
31116	164	148	120	0	0	28
31122	18	34	60	68	66	0
31128	0	28	34	24	4	68
31134	56	0	0	62	8	8
31140	16	16	32	0	0	17
31146	34	34	68	68	56	0
31152	0	34	34	36	36	40
31158	16	0	0	33	33	66
31164	66	90	36	0	0	34
31170	20	24	56	68	130	0
31176	0	34	20	8	16	32
31182	64	0	0	62	4	8
31188	16	32	124	0	24	16
31194	16	16	16	16	24	0
31200	120	64	32	16	8	4
31206	2	0	48	16	16	16
31212	16	16	48	0	0	16
31218	56	84	16	16	16	0
31224	0	0	0	0	0	0
31230	0	255	8	28	50	120
31236	48	50	124	0	0	0
31242	12	2	60	68	56	0
31248	0	16	16	60	34	66
31254	124	0	0	0	28	32
31260	32	64	56	0	0	2
31266	2	28	36	68	56	0
31272	0	0	28	34	124	64
31278	56	0	0	6	8	12
31284	16	16	32	0	0	0
31290	14	18	34	60	4	120
31296	0	16	16	62	34	68
31302	68	0	0	4	0	24
31308	8	16	120	0	0	2
31314	0	4	4	8	72	48
31320	0	16	20	56	48	72
31326	66	0	0	8	16	16
31332	32	32	24	0	0	0
31338	54	73	73	146	146	0
31344	0	0	60	34	34	68
31350	68	0	0	0	28	34
31356	36	68	56	0	0	0
31362	28	18	34	60	64	64
31368	0	0	30	18	36	60
31374	8	30	0	0	14	16
31380	16	32	32	0	0	0
31386	30	32	24	4	120	0
31392	0	4	30	0	16	18
31398	12	0	0	0	34	34
31404	68	68	56	0	0	0
31410	34	34	36	40	16	0
31416	0	0	65	65	146	146
31422	100	0	0	0	34	20
31428	24	40	68	0	0	0
31434	18	36	60	8	16	96
31440	0	0	120	8	16	32
31446	120	0	14	8	8	56
31452	8	8	14	0	8	8
31458	8	8	8	8	8	0
31464	112	16	16	28	16	16
31470	112	0	0	20	40	0
31476	0	0	0	0	60	66
31482	153	161	161	153	66	60

DATA

30720	0	0	0	0	0	0
30726	0	0	0	24	24	24
30732	24	24	0	24	0	36
30738	36	0	0	0	0	0
30744	0	36	126	52	52	126
30750	36	0	0	16	126	80
30756	126	22	126	16	0	102
30762	100	8	16	38	102	0
30768	0	56	40	16	42	100
30774	42	16	0	48	48	64
30780	0	0	0	0	0	24
30786	48	32	32	32	48	24
30792	0	24	12	4	4	4
30798	12	24	0	102	60	24
30804	24	60	102	0	0	24
30810	24	126	126	24	24	0
30816	0	0	0	0	56	56
30822	56	96	0	0	0	0
30828	126	126	0	0	0	0
30834	0	0	112	112	112	0
30840	0	24	24	0	126	0
30846	24	24	0	126	70	74
30852	82	98	126	0	0	24
30858	24	24	8	8	8	0
30864	0	126	70	6	126	64
30870	126	0	0	126	6	6
30876	126	2	126	0	0	124
30882	68	100	100	126	4	0
30888	0	126	66	64	126	6
30894	126	0	0	126	98	96
30900	126	66	126	0	0	126
30906	2	2	6	6	6	0
30912	0	126	98	98	126	66
30918	126	0	0	126	70	70
30924	126	2	2	0	0	24
30930	24	0	0	24	24	0
30936	0	24	24	0	0	24
30942	24	96	0	0	8	16
30948	48	16	8	0	0	0
30954	126	126	0	126	126	0
30960	0	0	8	4	6	4
30966	8	0	0	126	70	30
30972	24	24	0	24	0	127
30978	65	79	75	111	96	127
30984	0	60	36	62	98	98
30990	98	0	0	124	68	126
30996	98	98	126	0	0	126
31002	66	64	96	98	126	0
31008	0	126	66	66	98	98
31014	126	0	0	126	64	126
31020	96	96	126	0	0	126
31026	64	126	96	96	96	0
31032	0	126	66	64	102	98
31038	126	0	0	66	66	126
31044	98	98	98	0	0	16
31050	16	16	24	24	24	0
31056	0	4	4	4	6	70
31062	126	0	0	68	68	126
31068	98	98	98	0	0	64
31074	64	64	96	96	124	0
31080	0	126	74	74	106	106
31086	106	0	0	126	66	66
31092	98	98	98	0	0	126
31098	70	66	66	66	126	0
31104	0	126	66	126	96	96
31110	96	0	0	126	66	66
31116	66	78	126	0	0	120
31122	72	126	98	98	98	0
31128	0	126	64	126	6	6
31134	126	0	0	126	16	16
31140	24	24	24	0	0	66
31146	66	66	98	98	126	0
31152	0	98	98	98	38	36
31158	60	0	0	74	74	74
31164	106	106	126	0	0	66
31170	126	24	126	98	98	0
31176	0	66	66	126	24	24
31182	24	0	0	126	12	8
31188	16	48	126	0	0	30
31194	24	24	24	24	30	0
31200	0	96	48	24	12	6
31206	3	0	0	120	24	24
31212	24	24	120	0	0	24
31218	60	126	90	24	24	0
31224	0	0	0	0	0	255
31230	255	0	0	62	50	32
31236	112	32	126	0	0	0
31242	62	2	62	50	62	0
31248	0	64	64	124	68	100
31254	124	0	0	0	62	34
31260	32	50	62	0	0	2
31266	2	62	34	38	62	0
31272	0	0	62	50	62	32
31278	62	0	0	28	16	56
31284	16	24	24	0	0	0
31290	62	34	50	62	2	62
31296	0	32	32	62	34	50
31302	50	0	0	8	0	8
31308	8	12	12	0	0	8
31314	0	8	12	12	4	28
31320	0	72	88	80	124	100
31326	100	0	0	24	8	8
31332	8	12	12	0	0	0
31338	126	74	106	106	98	0
31344	0	0	62	34	34	50
31350	50	0	0	0	62	38
31356	34	34	62	0	0	0
31362	62	34	50	62	32	32
31368	0	0	62	34	50	62
31374	2	2	0	0	60	36
31380	32	48	48	0	0	0
31386	62	32	62	6	62	0
31392	0	16	56	16	16	24
31398	28	0	0	0	34	34
31404	34	50	62	0	0	0
31410	50	50	22	20	28	0
31416	0	0	66	82	86	86
31422	126	0	0	0	36	36
31428	24	52	52	0	0	0
31434	50	50	34	62	2	62
31440	0	0	62	12	0	24
31446	62	0	0	14	8	48
31452	8	8	14	0	0	24
31458	24	24	24	24	24	0
31464	0	112	16	12	16	16
31470	112	0	0	6	12	24
31476	48	96	192	0	0	102
31482	60	126	126	60	102	0

ROTULACION

30720	0	0	0	0	0	0
30726	0	0	0	48	48	48
30732	48	48	0	48	0	36
30738	36	0	0	0	0	0
30744	0	36	126	36	36	126
30750	36	0	0	8	62	40
30756	62	10	62	8	0	98
30762	100	8	16	38	70	0
30768	0	16	40	16	42	68
30774	58	0	0	8	16	0
30780	0	0	0	0	0	4
30786	8	8	8	8	4	0
30792	0	32	16	16	16	16
30798	32	0	0	0	20	8
30804	62	8	20	0	0	0
30810	8	8	62	8	8	0
30816	0	0	0	0	0	8
30822	8	16	0	0	0	0
30828	62	0	0	0	0	0
30834	0	0	0	24	24	0
30840	0	6	12	24	48	96
30846	64	0	0	108	198	206
30852	214	230	198	108	0	56
30858	24	24	24	24	24	60
30864	0	92	142	14	28	34
30870	94	204	4	92	142	14
30876	28	14	142	92	0	12
30882	44	76	140	238	12	30
30888	0	62	0	64	108	6
30894	70	44	0	108	194	192
30900	220	206	198	108	0	126
30906	252	128	8	24	48	48
30912	0	108	198	198	108	198
30918	198	108	0	108	198	230
30924	118	6	134	108	0	0
30930	24	24	0	24	24	0
30936	0	0	24	24	0	24
30942	24	48	0	14	28	24
30948	24	24	28	14	0	0
30954	126	0	0	126	0	0
30960	0	112	56	24	24	24
30966	56	112	0	44	70	14
30972	28	24	0	24	0	0
30978	0	0	0	0	0	0
30984	0	24	28	44	38	94
30990	70	70	0	236	102	102
30996	108	102	102	236	0	46
31002	98	96	96	96	98	44
31008	0	232	108	102	102	102
31014	108	232	0	238	98	100
31020	108	100	98	238	0	238
31026	98	100	108	100	96	224
31032	0	110	226	224	224	238
31038	230	108	0	198	198	198
31044	222	198	198	198	0	120
31050	48	48	48	48	48	120
31056	0	30	12	12	12	108
31062	108	24	0	230	108	104
31068	124	110	102	230	0	240
31074	96	96	96	96	100	236
31080	0	198	110	118	102	166
31086	134	134	0	226	114	58
31092	92	78	78	230	0	108
31098	198	198	198	198	198	108
31104	0	236	102	102	108	96
31110	96	240	0	108	198	198
31116	198	214	204	102	0	236
31122	102	102	120	108	108	230
31128	0	110	194	112	60	14
31134	134	236	0	180	180	48
31140	48	48	48	120	0	238
31146	196	196	196	196	204	88
31152	0	206	196	228	100	116
31158	56	24	0	218	218	218
31164	236	236	108	108	0	198
31170	194	112	56	28	44	198
31176	0	198	228	112	56	24
31182	24	24	0	230	142	28
31188	56	112	226	206	0	14
31194	8	8	8	8	14	0
31200	0	0	64	32	16	8
31206	4	0	0	112	16	16
31212	16	16	112	0	0	16
31218	56	84	16	16	16	0
31224	0	0	0	0	0	0
31230	0	255	0	28	34	120
31236	32	32	126	0	0	0
31242	12	2	60	68	56	0
31248	0	16	16	60	34	66
31254	124	0	0	0	28	32
31260	32	64	56	0	0	2
31266	2	28	36	68	56	0
31272	0	0	28	34	124	64
31278	56	0	0	6	8	12
31284	16	16	32	0	0	0
31290	14	18	34	60	4	120
31296	0	16	16	62	34	68
31302	68	0	0	4	0	24
31308	8	16	120	0	0	2
31314	0	4	4	8	72	48
31320	0	16	20	56	48	72
31326	66	0	0	8	16	16
31332	32	32	24	0	0	0
31338	54	73	73	146	146	0
31344	0	0	60	34	34	68
31350	68	0	0	0	28	34
31356	36	68	56	0	0	0
31362	28	18	34	60	64	64
31368	0	0	30	18	36	60
31374	8	30	0	0	14	16
31380	16	32	32	0	0	0
31386	30	32	24	4	120	0
31392	0	4	30	0	16	18
31398	12	0	0	0	34	34
31404	68	68	56	0	0	0
31410	34	34	36	40	16	0
31416	0	0	65	65	146	146
31422	100	0	0	0	34	20
31428	24	40	68	0	0	0
31434	18	36	60	8	16	96
31440	0	0	120	8	16	32
31446	120	0	0	14	8	49
31452	8	8	14	0	0	8
31458	8	8	8	8	8	0
31464	0	112	16	12	16	16
31470	112	0	0	20	40	0
31476	0	0	0	0	60	66
31482	153	161	161	153	66	60

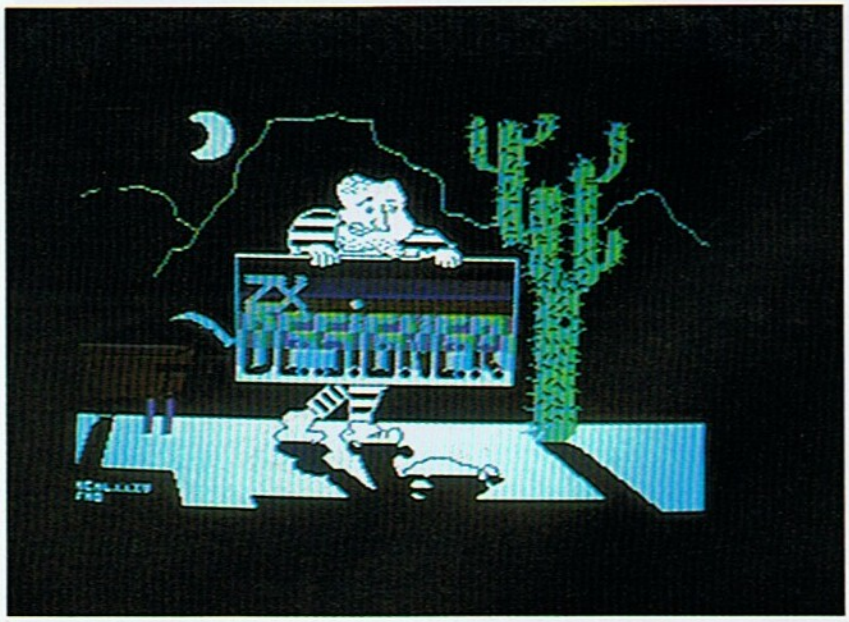
DISEÑADORES DE PANTALLAS

COMO ya hemos comentado anteriormente, incluso dedicándole un capítulo entero al tema, la estética es uno de los puntos fuertes en el éxito de cualquier programa. A nadie se le oculta que ésta se concreta principalmente en la pantalla, debido a lo cual, gran parte de los esfuerzos de programación se consumen en crear pantallas atractivas. Como quiera que ésto no es en muchas ocasiones tarea fácil, los programadores han creado herramientas que les hagan algo más sencillo su trabajo: los diseñadores de pantallas. Entre los diseñadores de pantallas existen lógicamente programas más o menos depurados, que se ajustarán a nuestro objetivo según lo complicado que sea éste. En general, un diseñador de pantallas es un programa que nos permite dirigir cómodamente por una pantalla limpia un punto que va dejando un rastro, a modo de lápiz.
Inevitablemente, y dado que la simple utilidad de ésto es bien poca, se le van añadiendo otras características encaminadas a mejorar la calidad estética del resultado. Así, el lápiz es capaz de pintar en cualquiera de los colores del Spectrum, borrar, invertir, o pasar de un punto a otro sin afectar a lo ya escrito.

ZX DESIGNER (ABC SOFT)

Las características de estos programas se van complicando, aportando cada vez más herramientas para facilitar el trabajo de diseño. Por ejemplo, el programa ZX DESIGNER, además de las cualidades comunes mencionadas, incorpora las siguientes posibilidades:
— Fijar un punto que se utilizará como base en las opciones de trazado de figuras que a continuación veremos.
— Trazar un arco, para lo cual bastará con pulsar la tecla **A** e introducir la longitud del arco en radianes.
— Trazar un paralelogramo.
— Trazar una línea hasta la «punta» de nuestro lápiz.
— Trazar un círculo, estableciendo como centro el cursor.
— Copiar la pantalla en la impresora.
— Cambiar los atributos con los que se dibuja en la pantalla.
— Rellenar de color cualquier figura.
— Guardar la configuración de la pantalla actual en la memoria.
— Sombrear una figura, mediante el relleno con un trazo discontínuo.
— Variar el paso con que el «lápiz» se desliza por la pantalla, para así poder alterar su velocidad de desplazamiento.
— Cuadricular la pantalla, que será una función muy útil para determinar los atributos de cada posición de carácter.
— Borrar la pantalla.
— Grabar o cargar la pantalla en cinta.
— Hacer más grueso el trazo del «lápiz».
— Definir caracteres gráficos (UDG).
— Cambiar de banco de caracteres definidos (dispone de ocho bancos).
— Grabar o cargar UDGs.
— Escribir o cargar UDGs.

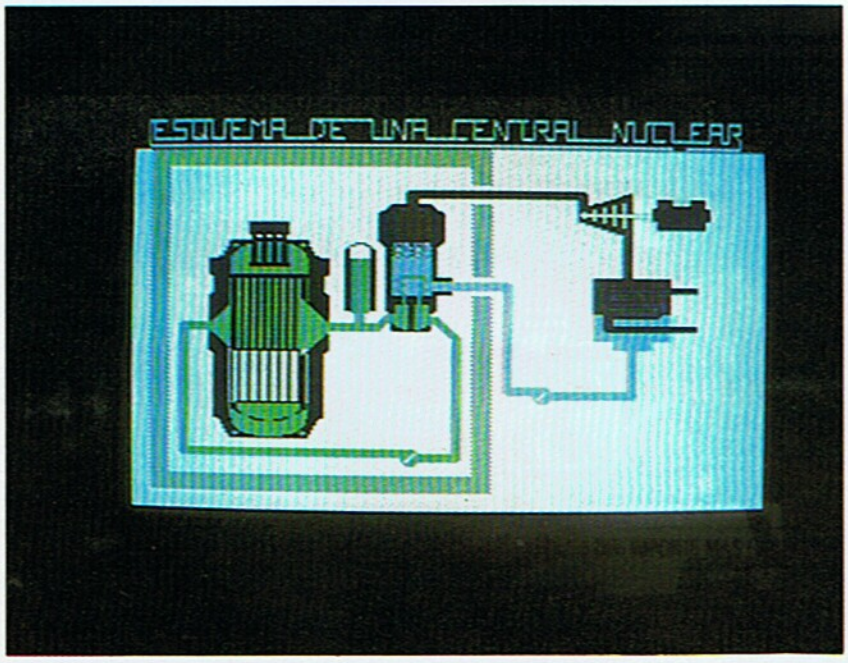
Aunque la pantalla de presentación del ZX Designer no es especialmente espectacular, con este programa podremos conseguir buenos resultados.



— Escribir textos en la pantalla.
— Borrar la última operación efectuada, función de extremada utilidad para los que somos demasiado ligeros de dedos.
— Ampliar una zona de pantalla.
— No alterar los atributos al paso del cursor.

Como podemos comprobar es muy amplia la gama de efectos que se puede obtener con este programa. Pero existen aún ciertas exigencias que quedan sin cubrir, como por ejemplo la confección de figuras tridimensionales.

El ARTIST cuenta entre sus características más destacables con un generador de gráficos definibles (UDG).



También la realización de gráficos técnicos puede ser contemplada por un diseñador de pantallas.

VU-3D (INVESTRONICA)

VU-3D es un sofisticado programa para dibujo tridimensional. Utilizando comandos sencillos podrá crear un objeto sólido, o conjunto de ellos, en el espacio tridimensional. Con este programa se puede obtener una imagen tridimensional del mundo haciendo rotar los objetos mirándolos desde diferentes ángulos.

Este mundo tridimensional se trata en conjunto con sus estructuras de datos, del mismo modo que una cámara programable, en el mismo sentido en que el ojo observa el mundo tridimensional. Podemos rodear un objeto y mirarlo desde lejos o de cerca desde diferentes direcciones. En este sentido, podemos diseñar un objeto sólido y obtener una sensación de su realidad tridimensional.

Para la confección de pantallas de este género contamos con la colaboración de los DISEÑADORES DE PANTALLAS.

La función de creación permite, mediante un sencillo juego de comandos, construir un objeto en tres dimensiones. Una vez construido éste, existe la posibilidad de acercarlo o alejarlo en un efecto de *zoom* o girarlo en cualquier dirección y sentido; estas operaciones posteriores a la creación son efectuadas en una segunda fase denominada *display* (pantalla).

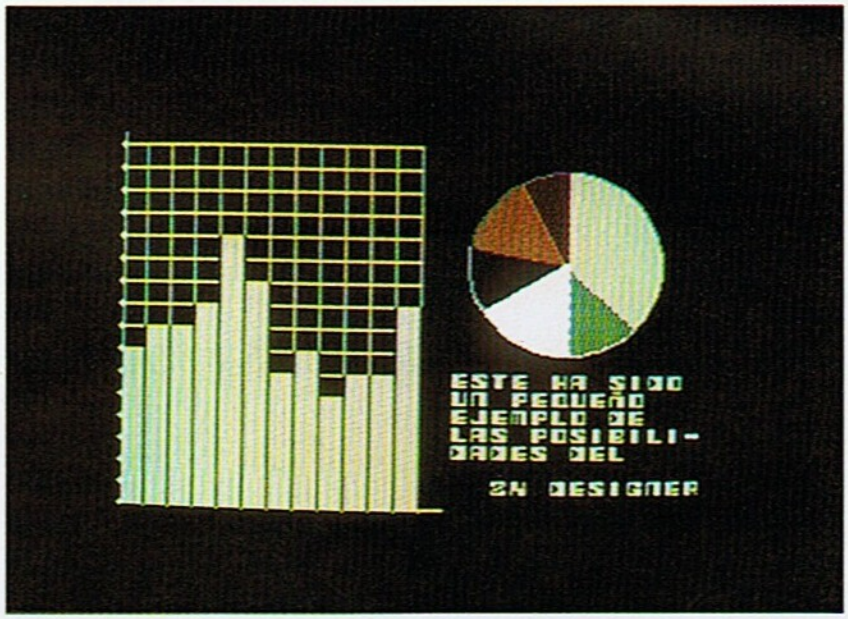
En una tercera etapa, habiéndose creado un objeto en la fase de creación y fijada su ampliación y punto de observación en la de *display*, podemos obtener, en un color de nuestra elección, el objeto en sus otras dos formas de presentación: *hidden line* (superficie frontal) o *shade* (sombreado). Esta tercera etapa es la denominada *picture* (imagen).

Las opciones de la fase *picture* son: *shade* (sombra), *hidden line* (superficie frontal), *print* (impresora), *colour* (color), *keep* (guardar), *quit* (salir). La función *colour* permite elegir el color apropiado en cada caso a la imagen representada, *print* realiza una copia en papel y *quit* nos devuelve a la fase *display*.

keep es una opción muy a tener en cuenta, puesto que nos permite grabar la figura como una pantalla, la cual podrá ser cargada utilizando **SCREEN$** sin la presencia del programa VU-3D.

El comando *hidden line* produce una imagen frontal de los objetos existentes en pantalla. Los extremos que limitan las caras superiores del objeto, que son aquellas que el observador puede ver, son las únicas líneas dibujadas, mientras que los de la cara posterior no se dibujan. La ejecución de este comando puede tardar un tiempo considerable si se trata de una figura complicada con muchas caras.

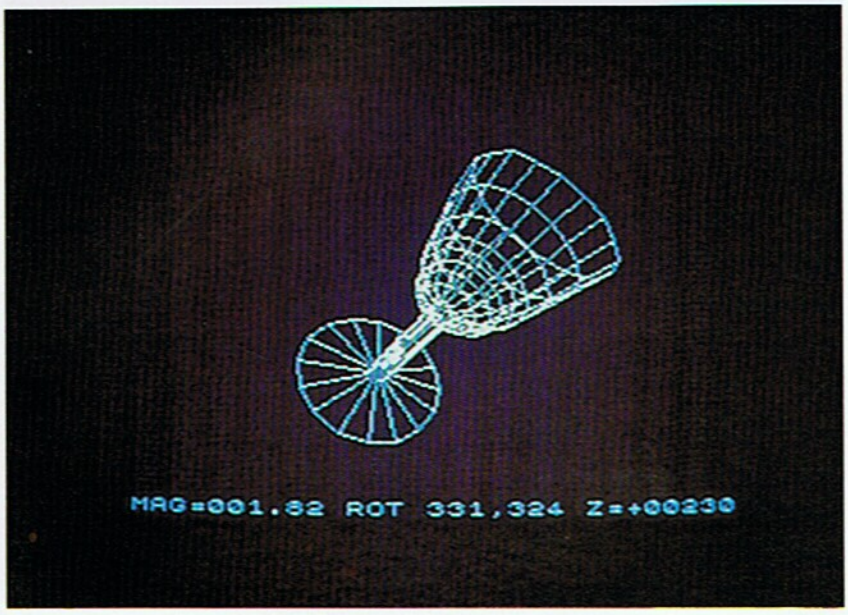
Otra forma de ver el objeto u objetos de forma tri-



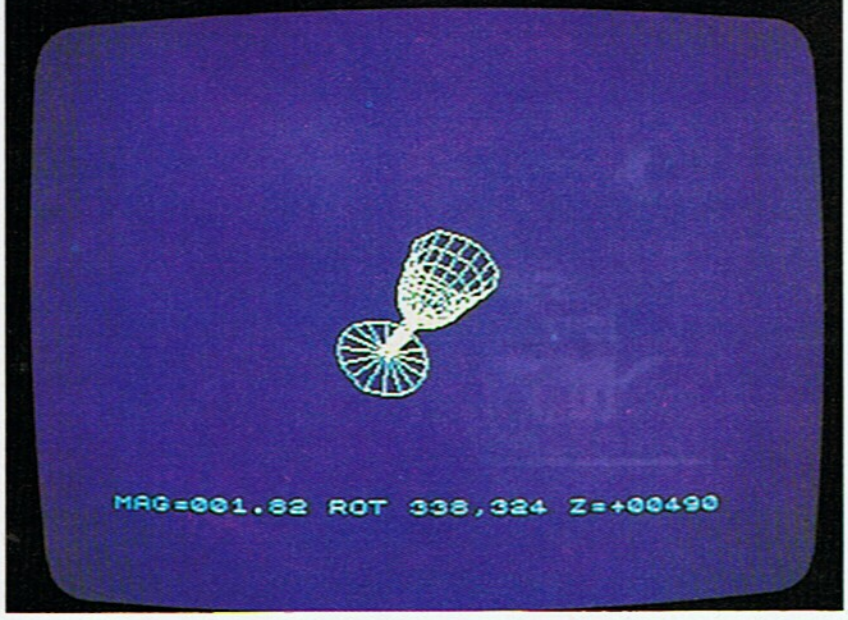
El diseño de pantallas no sólo se emplea en el ámbito de los juegos, sino también en el comercial.

La cara B de la cinta de ZX Designer se encabeza con esta pantalla de presentación, seguida del programa en sí. La cara A está destinada a demostraciones.

dimensional es sombrear aquellas caras que el observador puede ver. La claridad del sombreado de cada cara viene determinada por una combinación de luz de fondo y reflexión difusas desde una única fuente luminosa.
Así, utilizando el comando *shade* se nos interrogará sobre dónde deseamos situar la fuente de luz: arriba, centro o abajo, y además, izquierda, centro o derecha.
Además de las etapas mencionadas el programa nos permite modificar la figura diseñada o grabar, y por supuesto a cargar, las características de la figura definida, no en forma de pantalla sino de fichero de datos. Finalmente, la opción *abandon* (abandonar) nos permite borrar el fichero de datos existente, dejando la memoria libre para una carga desde casete, o crear una nueva figura en la primera etapa.



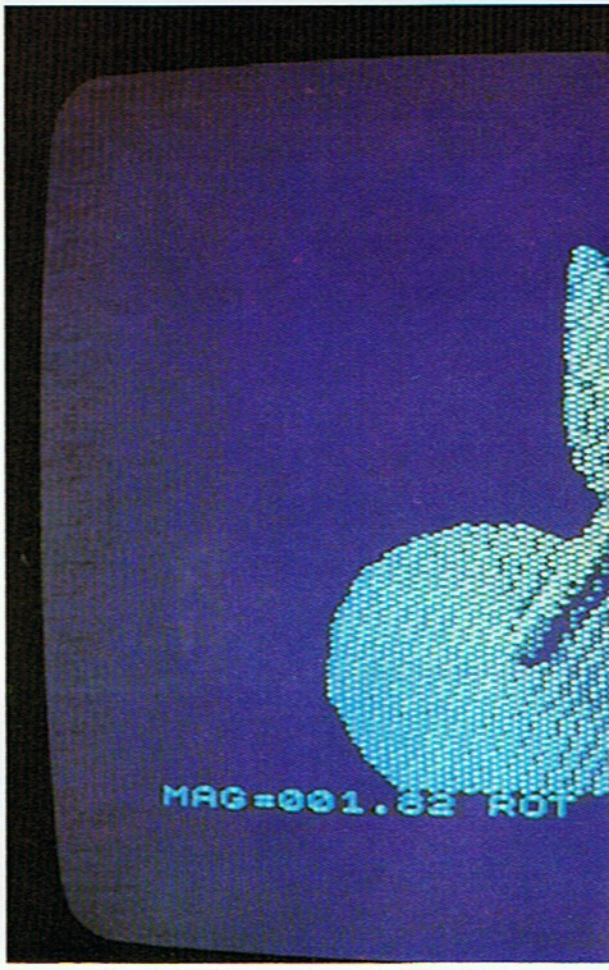
El programa VU-3D nos permite obtener una imagen tridimensional, haciendo rotar los objetos y mirando desde diferentes ángulos.



Al igual que la ampliación, VU-3D permite también la reducción de la imagen dibujada.

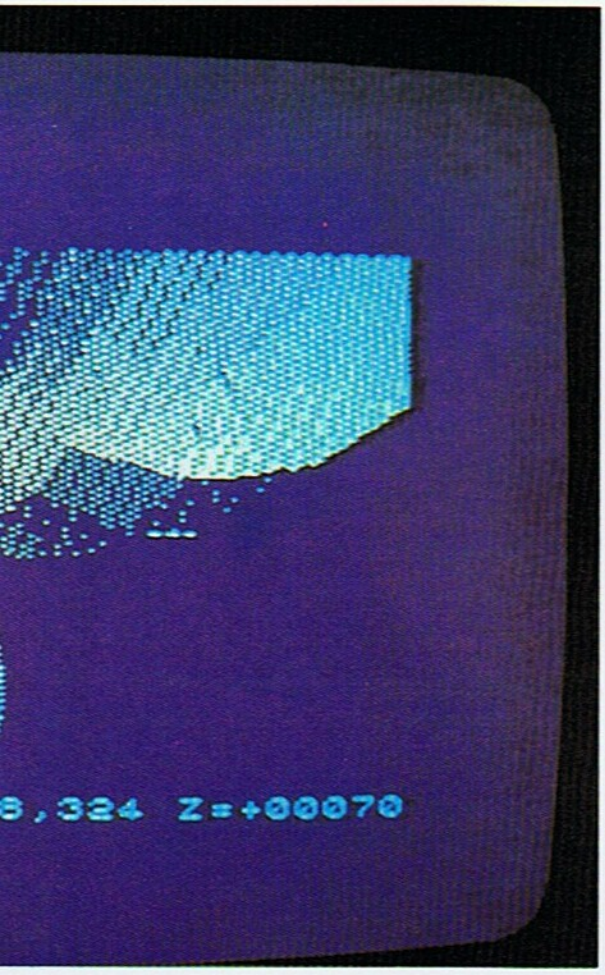
ARTIST (DINAMIC)

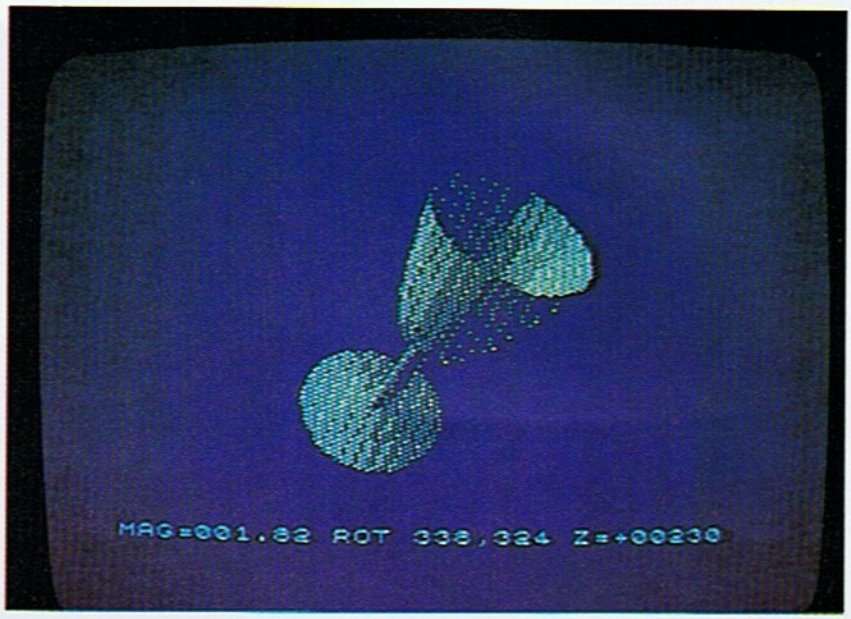
ARTIST es, sin duda, uno de los más completos equipos gráficos para el diseño de pantallas presentes en el mercado nacional. Su acción se concreta en tres puntos fundamentales: creación de figuras en tres dimensiones, definición de caracteres UDGs y diseño sobre la pantalla de alta resolución.



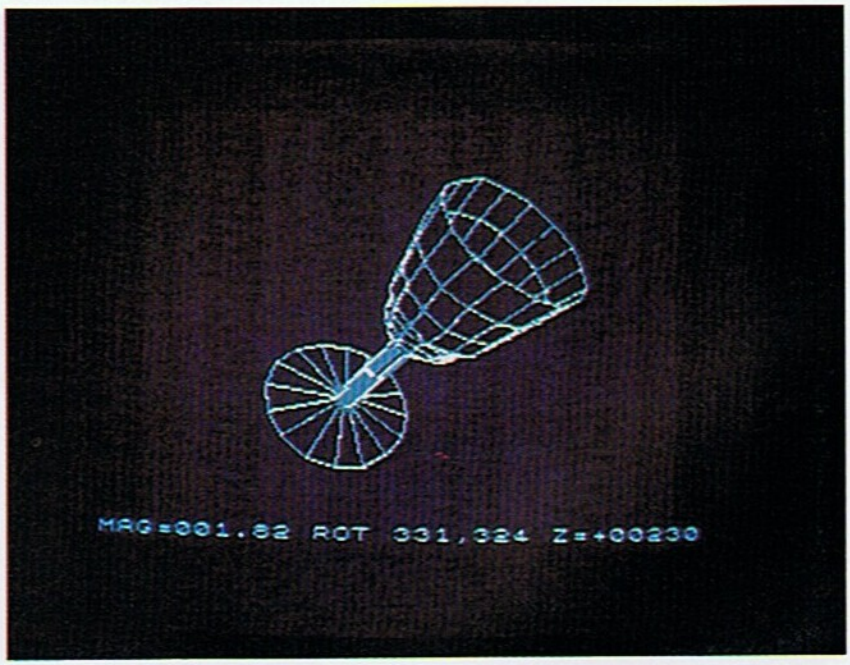
Las diversas opciones del programa son combinables: en el ejemplo se ha empleado el sombreado y la ampliación.

En el denominado modo *plotter* se nos presenta una pantalla libre de 176 por 256 *pixels* en la cual podemos desplazar un cursor dejando un rastro a su paso, permitiendo deslizar dibujos sobre la pantalla. Este rastro vendrá dado en el color que nosotros indiquemos, pudiendo ser alterado éste en cualquier momento mediante la pulsación de las teclas numéricas del 0 al 7.
De modo similar podemos alterar el color de fondo y también de dos características tan importantes como el **FLASH** y el **BRIGHT.** Asimismo, también es posible definir el tamaño del cursor (pequeño o grande), o la velocidad de desplazamiento del mismo por la pantalla (tres velocidades). Naturalmente, también podemos desplazar el cursor borrando en vez de escribiendo, o simplemente dejando inalterado el contenido de la pantalla.
El programa también prevé la posibilidad de trazar líneas rectas, arcos circulares, e incluso círculos, representando sin duda una gran ayuda a la hora de confeccionar este tipo de figuras de cierta dificultad sin pretender ser realizadas a mano alzada.
Asimismo, la posibilidad de hacer *fill* (rellenar una superficie cerrada de un color determinado), acompañada del establecimiento de un *grid,* conforman las características más destacables de este modo. El *grid* es una cuadrícula de colores que no afectan al dibujo pero nos ayudan, aprovechando la posibilidad de realizar *scrolls* de la pantalla, a realizar una correcta utilización del color en cada posición de carácter.
En este modo el programa ARTIST es muy similar a MELBOURN DRAW que más adelante pasaremos a estudiar.
Sobre esta misma pantalla también es posible es-





Una de las opciones más interesantes del programa es la de sombreado, pudiendo elegir incluso la situación del foco de luz.



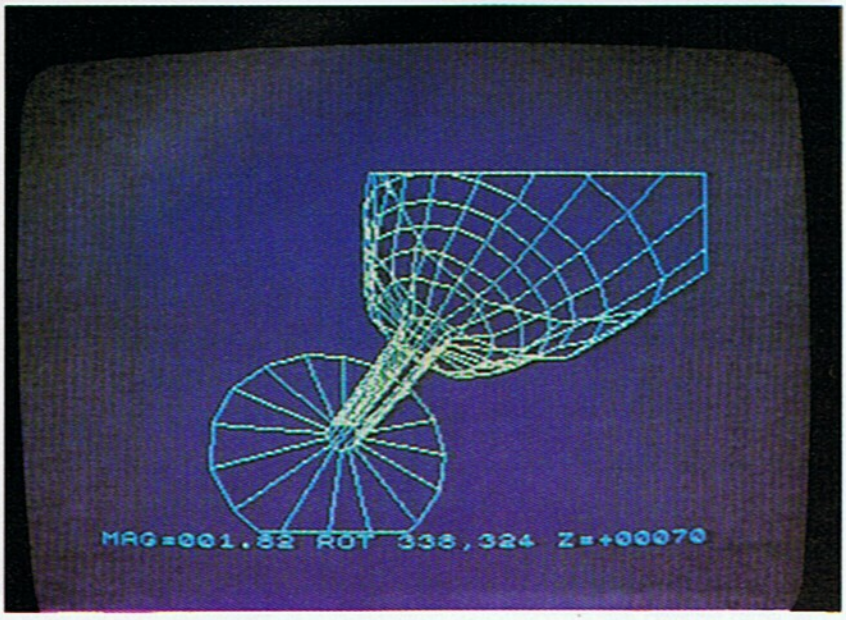
La opción de hidden line no efectua la representación de las lineas ocultas del objeto.

cribir texto, incluso en caracteres gigantes, introduciendo el mensaje y el tamaño deseado para las letras.

En cuanto a la creación de gráficos definidos, disponemos de un clásico generador de UDGs muy similar a otros de este tipo, como el que acompaña la cinta de demostración de nuestro aparato. Así por ejemplo nos permite:

— Definir el gráfico en el carácter que queramos del banco seleccionado.
— Invertir el gráfico.
— Girar el gráfico a la derecha.
— Reflejar el gráfico.
— Borrar el grafico.

El Artist también incorpora un tratamiento de imágenes tridimensionales, que podemos fundir una vez confeccionada con la pantalla principal. Para su diseño empleamos puntos definidos por tres coordenadas (X, Y y Z), que podemos dejar libres, o unir mediante una línea con el punto anteriormente definido. De forma similar al VU-3D, es posible acercar y alejar la figura, así como girarla en cualquier dirección y sentido.



VU-3D contempla la posibilidad de ampliación de las imágenes.

En el primer nivel de ZOOM la pantalla se muestra multiplicada por cuatro.

MELBOURNE DRAW (MELBOURNE HOUSE)

Para finalizar vamos a hablar de un tipo de diseñador de tipo medio, que incorpora funciones si-

El MELBOURNE DRAW incorpora funciones muy similares al modo plotter del ARTIST.

milares al modo *plotter* del Artist, y tiene como característica fundamental su sencillez de manejo: el MELBOURNE DRAW.

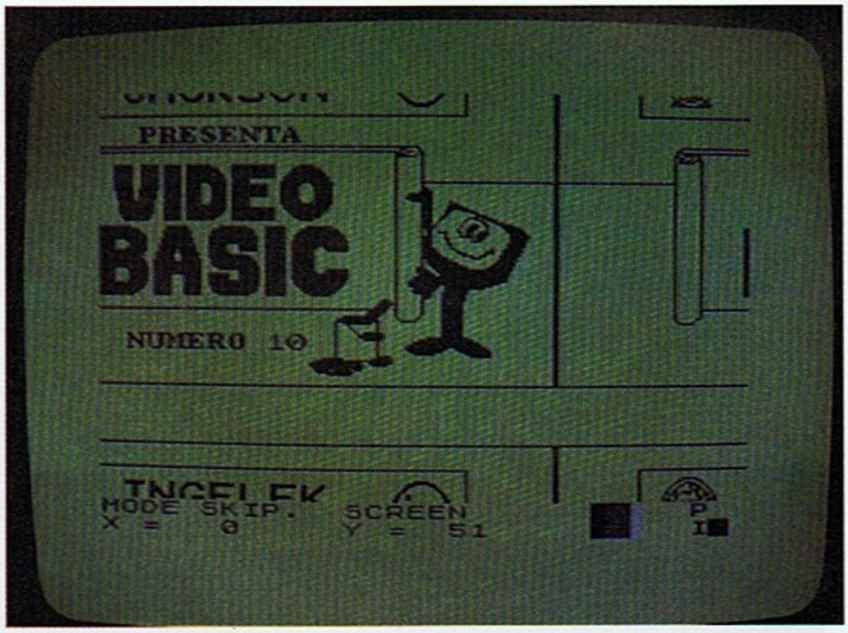
El programa se compone de un pequeño soporte BASIC, y un bloque de código máquina, que efectúa propiamente el trabajo de diseño gráfico. El BASIC tan solo contiene un menú de opciones que da acceso al modo de edición y se ocupa de gestionar la grabación, carga y verificación de las pantallas o juego de gráficos definidos creados.

Una copia de la pantalla se realiza sobre otra zona de memoria, de forma que podemos trabajar sin problema en las 24 líneas de la pantalla.

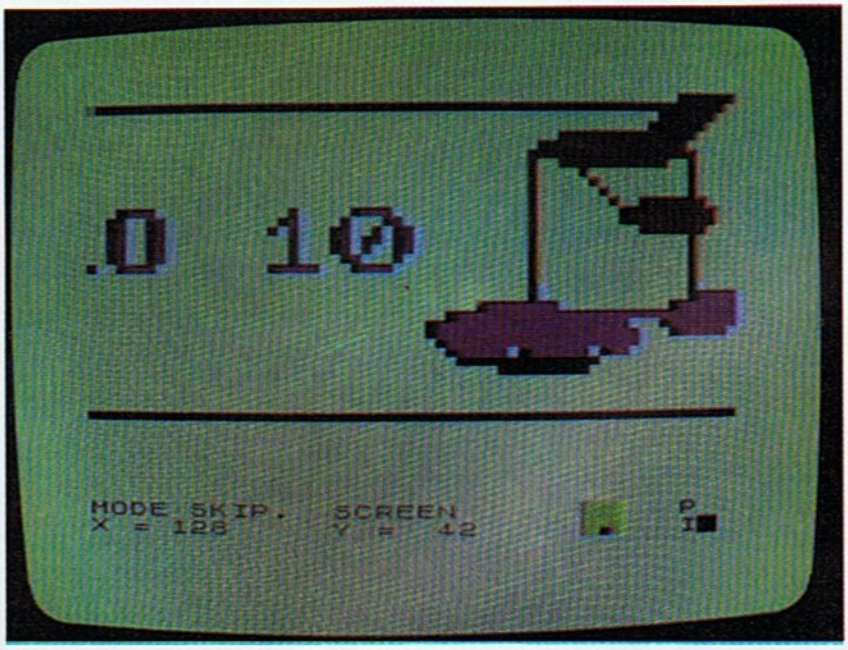
Al entrar en el modo edición, un cursor representado por un punto de *plot* intermitente indica nuestra posición.

Para el desplazamiento del punto, utilizaremos las teclas que circundan la **S** (Q, W, E, A, D, Z, X y C), pudiendo encontrarse el cursor en cuatro modos: SKIP, SET, RESET o INVERT. El primero de ellos se selecciona mediante la pulsación de **SPACE** y nos permite desplazarnos sin afectar a la pantalla; el segundo se activa mediante **ENTER** y hace que el punto vaya dejando rastro a su paso. El tercer modo (RESET) se utiliza para borrar y se activa con la tecla **O**, mientras que la tecla **I** sitúa el modo INVERT que cambia de estado la pantalla en los puntos por los que pasa el cursor, es decir, apaga los puntos encendidos, y enciende los apagados.

Aunque el desplazamiento del cursor es nuestra herramienta básica para el dibujo, disponemos también de otras funciones de gran interés. En primer lugar, la pantalla va siempre acompañada de una zona de indicadores de dos líneas, que indican el lugar de la pantalla en que se encuentra el cursor (coordenadas X, Y), el modo de dibujo



La posibilidad de scroll punto a punto y en cualquier sentido, es otra de las características más interesantes del MELBOURNE DRAW.



En el segundo y último nivel de ZOOM la pantalla aparece multiplicada por dieciséis.

(SKIP, SET, ETC...), los atributos empleados (fondo, tinta, brillo y parpadeo), y otras características que a continuación veremos. Esta zona, que habitualmente se sitúa en las dos últimas líneas de la pantalla (las reservadas al sistema), puede ser desplazada temporalmente a las dos primeras, pulsando **CAPS SHIFT + 9**, lo cual nos permite dibujar en las veinticuatro líneas de la pantalla.

El cambio de atributos se efectúa de una manera muy rápida; simplemente tenemos que pulsar la tecla numérica correspondiente al color deseado, sin **CAPS SHIFT** para la tinta y con **CAPS SHIFT** para el fondo, mientras que las teclas **CAPS SHIFT + V** afecta al modo **FLASH** y **CAPS SHIFT + B** a **BRIGHT.** Por otra parte, para hacernos una idea de conjunto, también podemos alterar el color del marco de pantalla, pulsando la tecla **B** y contestando a la pregunta **BORDER?** que aparecerá en la zona de indicadores con el código de color deseado.

Una de las propiedades del MELBOURNE DRAW es marcarnos, sin deteriorar la pantalla, la situación de las posiciones de carácter, para facilitar la asignación de color.

Además del aumento temporal (ZOOM), el MELBOURNE DRAW puede efectuar aumentos definitivos mediante la opción SCALE.

Para efectuar acciones sobre la pantalla completa, deberemos pulsar **CAPS SHIFT + R**, lo cual nos dará acceso a un menú en la zona de indicadores para situar todo el fondo de pantalla del color fijado para fondo (opción P del menú), toda la tinta del color de la tinta (opción I), asignar fondo y tinta a la vez (B), borrar la pantalla (S), borrar la pantalla y ajustar el color de fondo y tinta (A), o bien regresar a la pantalla principal (N).

La asignación de color no sólo se efectúa en el momento de trazar el dibujo, sino que también puede ser establecida independientemente pulsando **H.** El cursor pasará a tener las dimensiones de un carácter, funcionando en los modos SET o SKIP, aunque en esta ocasión, en el modo SET no producirá modificación de la pantalla sino de los colores, tal como son establecidos mediante las teclas numéricas.

Para ayudarnos en la tarea de colorear, disponemos de un grid para orientarnos claramente sobre las posiciones de carácter en las que se encuentran los elementos del dibujo (tecla **G**) y de un sistema de *scroll* que nos puede ayudar a cuadrar las figuras en los caracteres que nos interesen; para ello pulsaremos la tecla **K** y desplazaremos la pantalla utilizando los mismos controles que para el cursor (teclas de alrededor de la **S**). Finalmente, el **FILL** (rellenado de superficies) que se consigue con la pulsación de **CAPS SHIFT + F** es otra característica de interés.

En lo referente al texto, también puede ser incluido libremente en la pantalla, pasándose a este modo con la pulsación de **T.** En cuanto a los caracteres definidos, pueden ser también utilizados, disponiendo el modo **GRAPHICS (CAPS SHIFT + 9)** dentro del modo texto. La pulsación de **U** en el modo de pantalla principal, nos facilita la definición de cualquier UDG, con lo cual la pantalla se convierte en una enorme parrilla de definición.

Por último, existe la posibilidad de ampliar y reducir temporalmente el contenido de la pantalla (**M** y **N**), para mayor detalle, o bien ampliar o reducir la pantalla definitivamente **(CAPS SHIFT + 8)**.

En el primer caso, se trata simplemente de un ZOOM de la zona de la pantalla en que nos encontremos, que puede ampliar la imagen cuatro u ocho veces. En el segundo caso se trata de una modificación permanente para ampliar el dibujo de toda la pantalla, con la consiguiente pérdida de parte de la información.

ANALIZANDO LOS GRUPOS

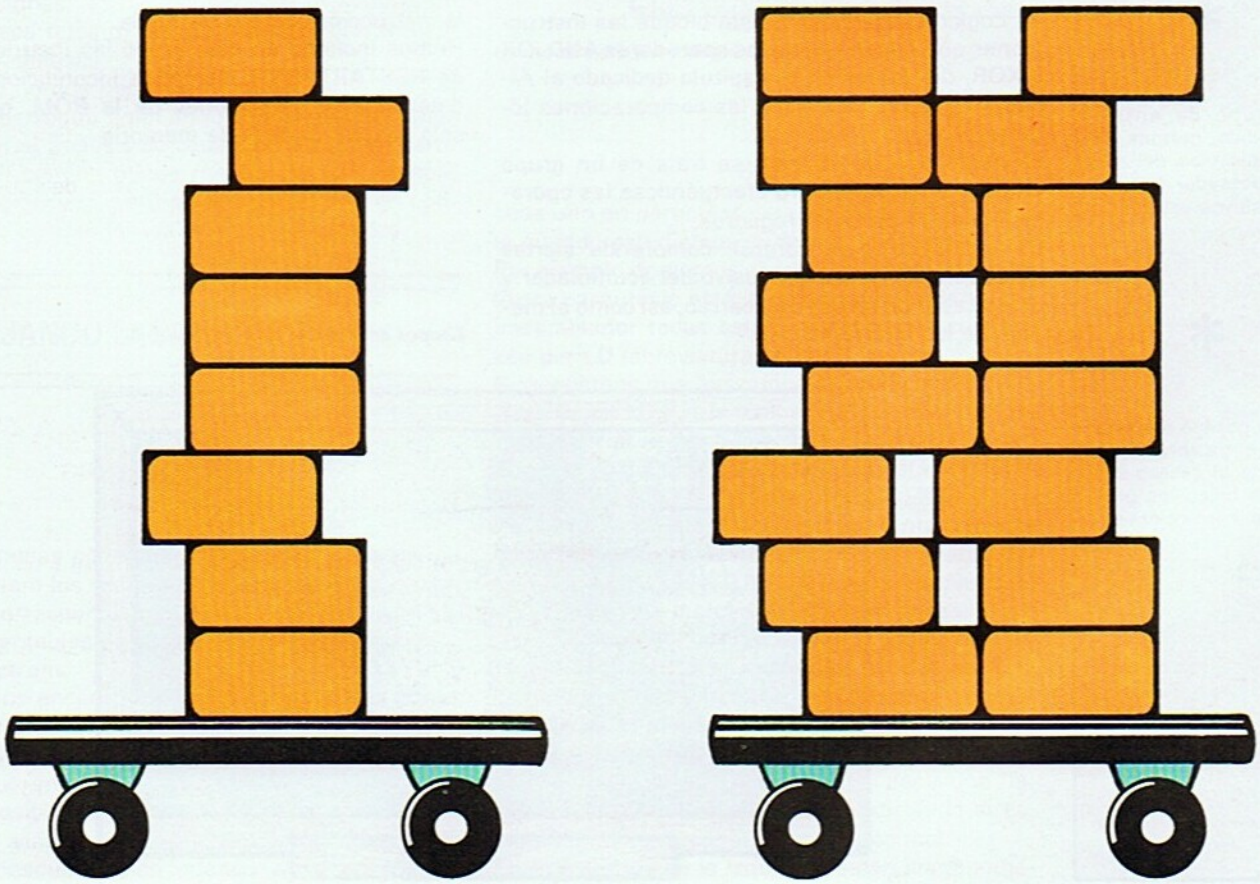
MUCHOS quizá nos estemos preguntando cuáles son las auténticas posibilidades del microprocesador de nuestro Spectrum. Somos bombardeados con una avalancha de nuevos conceptos y todavía no se ha justificado su utilidad.
Paciencia, fue necesario sentar unas sólidas bases antes de proseguir nuestro estudio del C/M. Efectuemos un recorrido a través de los diferentes grupos de instrucciones para hacernos una idea bastante aproximada de las operaciones que puede ejecutar el Z 80.

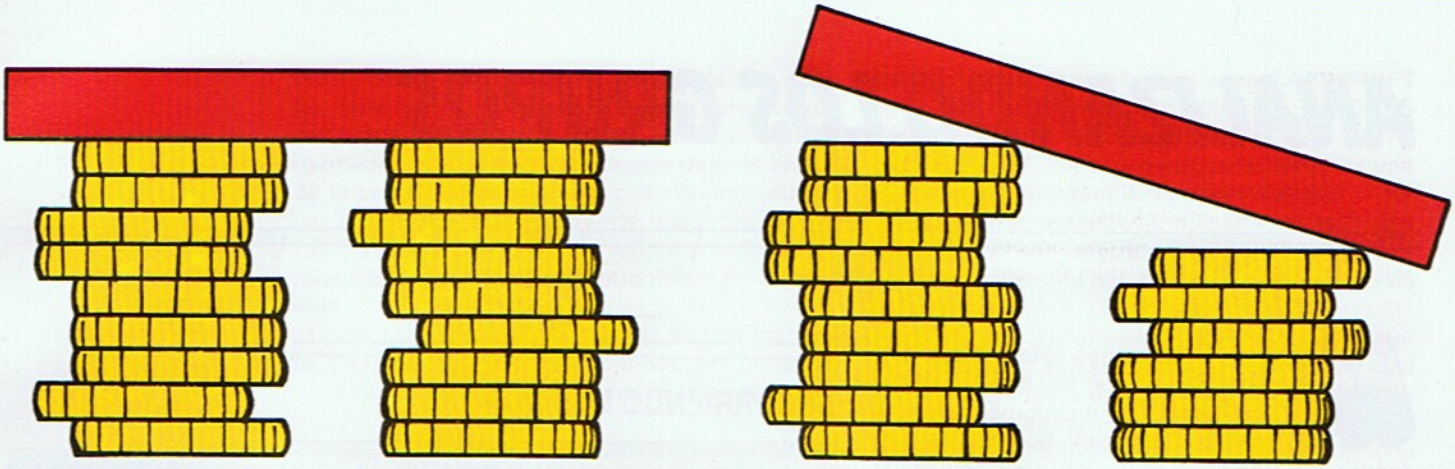
RECORRIENDO LAS TABLAS

Volvamos a las tablas del capítulo anterior. Allí encontraremos el juego de instrucciones completo del Z 80 (casi 700 instrucciones diferentes), las cuales hemos subdividido en los siguientes bloques:

1) Carga de 8 bits: en este grupo se hayan encuadradas todas las instrucciones de intercambio de información entre registros individuales, y entre registros y posiciones de memoria.

Grupos de carga de 8 y 16 bits.





Grupo lógico.

¡!

Las instrucciones de manipulación de bit constituyen el más numeroso grupo entre las nuevas implementadas en el microprocesador Z 80.

*

El grupo de entrada/salida permite la comunicación del microprocesador con los dispositivos exteriores a él.

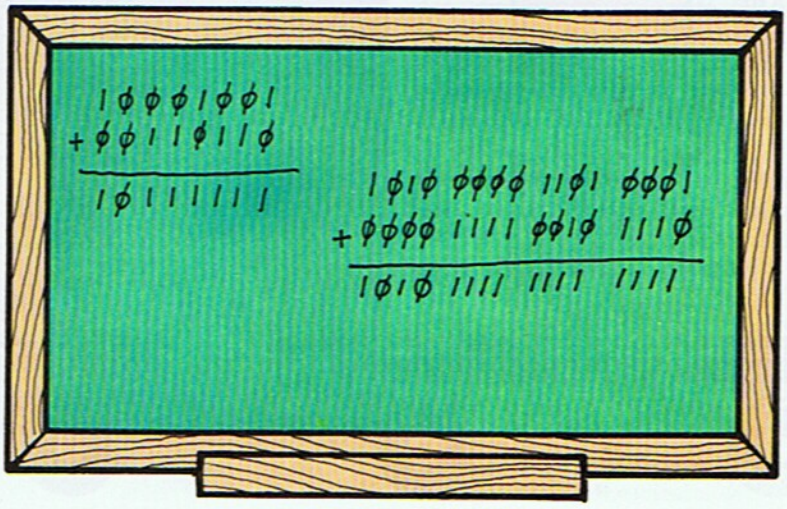
*

No es posible dar una norma general que permita el acceso común a todos los grupos de instrucciones.

2) Carga de 16 bits: como observaréis, muy pocas transferencias de información pueden efectuarse entre pares de registros. Sin embargo, a este grupo pertenecen las instrucciones PUSH y POP, fundamentales en el manejo del STACK o pila.
3) Aritmético de 8 bits: las operaciones de suma y resta entre cantidades de 8 bits son ejecutadas por las instrucciones de este grupo. Es posible también, incrementar o decrementar en uno el contenido de un determinado registro o dirección de memoria.
4) Lógico: pertenecen a este bloque las instrucciones que hacen uso de los operadores AND, OR y XOR, discutidos en el capítulo dedicado al Algebra de Boole, así como las comparaciones lógicas.
5) Aritmético de 16 bits: se trata de un grupo análogo al anterior pero efectuándose las operaciones entre pares de registros.
6) Aritmético y de control: comprende ciertas instrucciones de uso exclusivo del acumulador y el indicador de arrastre o acarreo, así como el manejo de las interrupciones.

7) Saltos y llamadas: permiten que el control del programa sea transferido a otra zona de éste, incondicionalmente o en determinados supuestos establecidos previamente, volviéndose al punto donde se efectuó la bifurcación, en el caso de que la instrucción fuera de llamada.
Hemos incluido en este grupo las instrucciones de RESTART (RST), llamadas incondicionales a determinadas direcciones de la ROM, que tan solo ocupan un byte de memoria.

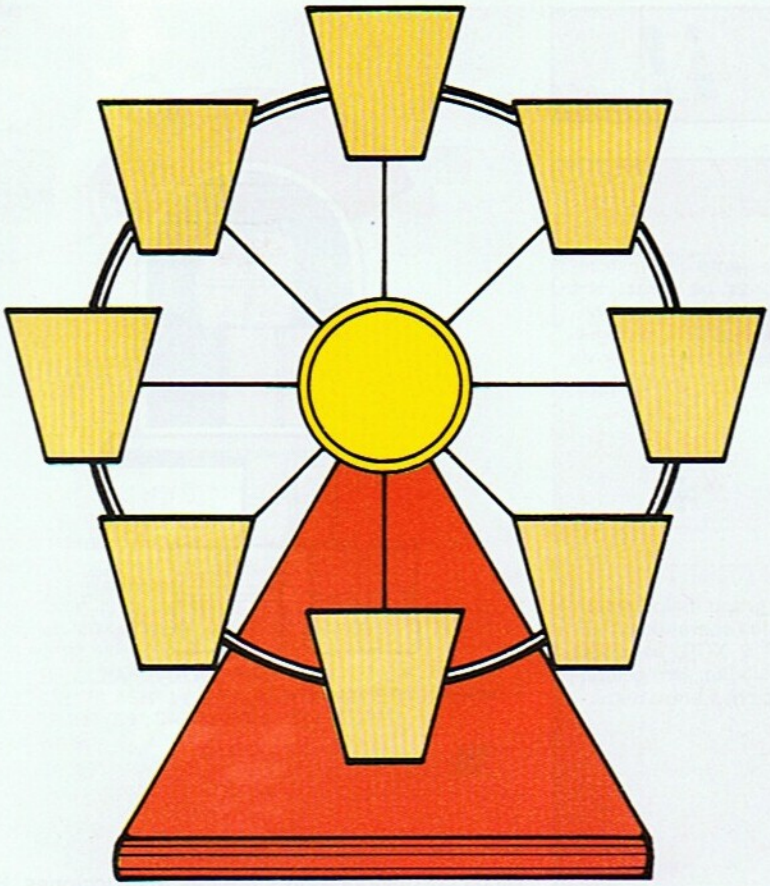
Grupos aritméticos de 8 y 16 bits.



8) Retornos: las instrucciones de retorno permiten que el programa continúe su ejecución a partir del punto donde fue efectuada la última llamada. De la misma manera que en el grupo anterior, el regreso puede efectuarse según se cumplan o no algunas condiciones especiales.
9) Intercambio: en este grupo encontramos las únicas instrucciones del microprocesador Z 80 que permiten el manejo del grupo de registros alternativos (ARS).
10) Transferencia: facilitan el movimiento de bloques de datos desde unas posiciones de memoria a otras.
11) Búsqueda: permiten examinar un bloque de memoria con la intención de hallar un byte con el mismo contenido que el almacenado en el acumulador.
12) Tratamiento de bits: 240 nuevas instrucciones las cuales permiten manipular individualmente cada uno de los bits almacenados en cualquier registro o posición de memoria.
13) Rotación y desplazamiento: al igual que las anteriores, posibilitan la ejecución de operaciones a nivel de bit, dentro de cada octeto (byte).
14) Entrada/Salida: estas instrucciones habilitan al microprocesador para la comunicación con los dispositivos exteriores a él, recogiendo o enviando información desde o hacia ellos.
Nuestra curiosidad debe estar ahora satisfecha. No dudaremos que con semejante volumen de instrucciones, las posibilidades de programación que ofrece nuestro microprocesador son francamente fabulosas.
Pero no hemos hecho sino comenzar, y todavía hemos de seguir ampliando conocimientos antes de pasar al análisis de cada instrucción en particular. El camino puede resultar largo y complicado a principio, pero pronto estaremos en condiciones de obtener los primeros resultados.

MANEJANDO LAS TABLAS

En las tablas podemos encontrar de manera rápida y clara los códigos de operación (en hexadecimal) correspondientes a todas las instrucciones implementables en el microprocesador de nuestro Spectrum.
Quizá nos encontremos algo sorprendidos por la anterior afirmación y no seamos capaces de ver en ellas otra cosa que un maremagnum de símbolos aparentemente sin sentido.
No es posible dar una norma general, la cual permita el acceso común a todos los grupos, pues como descubriremos cuando sean analizados



Grupo de rotación y desplazamiento.

cada uno en particular, su utilidad se hará patente cuando estudiemos su forma de operar.
Por ejemplo, tomemos el grupo de carga de 8 bits. Entre paréntesis está señalado que en lenguaje ensamblador todas estas instrucciones comienzan por LD (abreviatura de LoaD, cargar).
Supongamos que buscamos la codificación de la instrucción LD B,A la cual significa que introduzcamos en el registro B el contenido del registro A. En la parte superior de la tabla está señalada la palabra ORIGEN y a la izquierda DESTINO.
Por tanto, el destino es B y el remitente o fuente de la información es A. Entrando en la tabla por la fila señalada con B, buscamos la intersección con la columna marcada con A y en la casilla correspondiente encontramos que el código de operación asociado al ensamblador LD B,A es 47h.
En otros grupos se define directamente el mnemónico de la instrucción, como en el caso de los saltos y llamadas. Supongamos que la instrucción a codificar es CALL 7FFF, es decir una llamada a la subrutina ubicada a partir de la dirección 7FFF hexadecimal o 32767 decimal.
En la columna de la izquierda están los mnemó-

¡!

Mediante las instrucciones de carga, son transferidos datos entre registro, o registros, y posiciones de memoria.

*

El grupo lógico habilita los operandos AND, OR y XOR, así como posibilita las comparaciones entre datos.

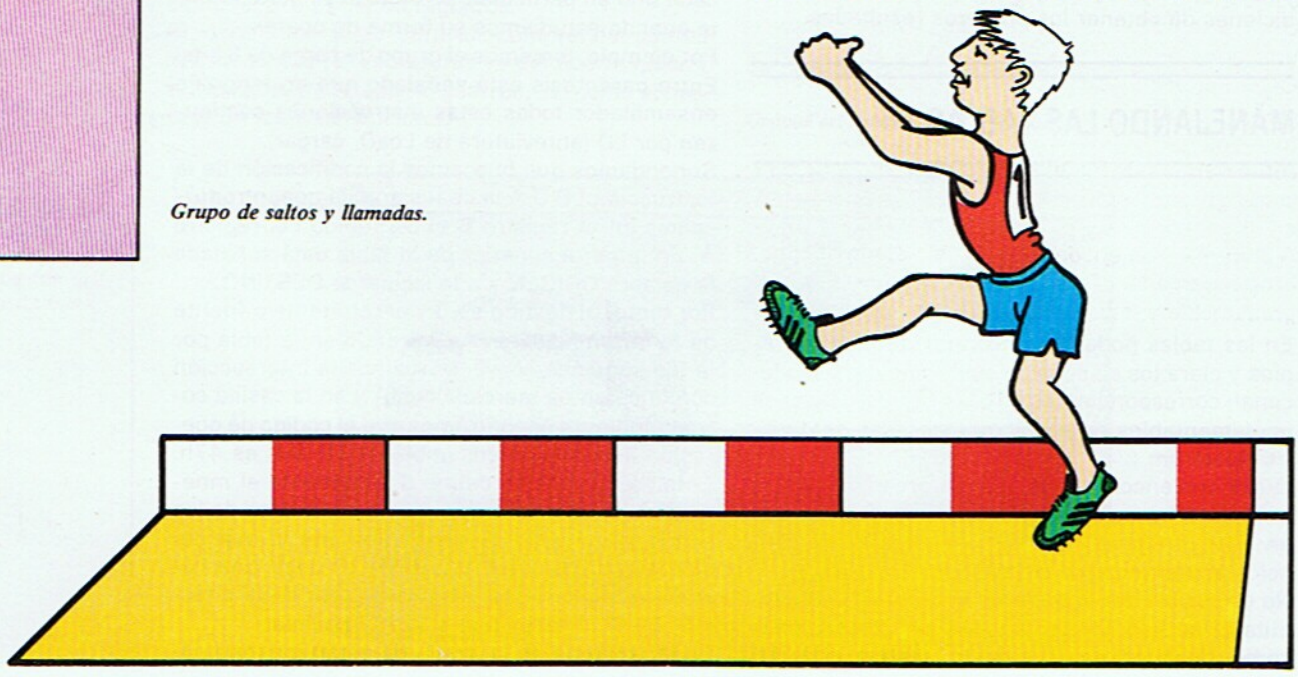
Grupo de transferencia.

nicos correspondientes a estas instrucciones. En la sexta fila encontramos CALL. La llamada ha sido incondicional a la dirección especificada, a continuación seguirá el formato CALL nn, siendo *nn* los dos bytes que definen dicha dirección.
Luego la codificación correcta será CD FF 7F. No nos sorprendamos si encontramos los dos bytes con su posición intercambiada. Ya aclaramos en su momento, que el microprocesador cuando se trata de una dirección espera encontrar en primer lugar siempre el byte de menor peso.
Otros símbolos utilizados son, por ejemplo, la «d» que representa el byte de desplazamiento en las instrucciones que manejan el direccionamiento indexado, o la «e», la cual representa la extensión del salto en el direccionamiento relativo (los modos de direccionamiento serán el tema principal de nuestro próximo capítulo).

CARGADOR DE C/M

Para terminar el presente capítulo, hemos preparado un pequeño programa, el cual actuará de cargador hexadecimal para todas las rutinas en C/M con las que pondremos en práctica los ejemplos necesarios al explicar los distintos grupos de instrucciones.
La elección del sistema hexadecimal nos ha parecido la más adecuada, dada la extensa bibliografía existente que lo utiliza, en especial la literatura dedicada a la descripción de las rutinas almacenadas en la ROM.
Los códigos hexadecimales correspondientes a las instrucciones que configuren nuestros programas estarán incluidos en las sentencias **DATA** que situaremos al principio del programa,

Grupo de saltos y llamadas.



y efectuando **RUN 9000**, quedarán almacenados en la memoria de nuestro ordenador para su posterior ejecución.

En caso de cometer algún error, se nos indicará la sentencia donde ocurrió para modificarla, mediante un mensaje intermitente que aparecerá sobre la pantalla del monitor.

Emplearemos el modo mayúsculas al introducir los caracteres hexadecimales que conformen nuestra rutina, siendo siempre los dos últimos de esta **, de modo que el programa reconozca que no ha de seguir almacenando más código en la memoria.

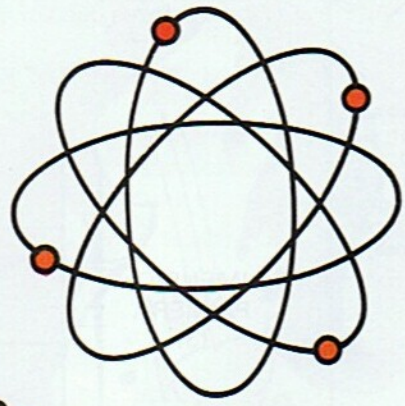
Al comienzo se nos preguntará sobre la posición de memoria donde deseamos introducir la rutina en C/M. En principio, toda la RAM podría utilizarse, pero para evitar posibles conflictos con las Variables del Sistema y otras zonas delicadas de éste, elegiremos posiciones de memoria cercanas al RAMTOP donde ejecutar nuestros programas, salvo que no se indique lo contrario.

Una vez almacenado el C/M es posible grabarlo utilizando el comando BASIC **SAVE "Nombre" CODE** dirección de inicio, longitud, siendo dirección de inicio la elegida por nosotros previamente y longitud, el número de bytes ocupados por la rutina (al finalizar la ejecución del programa cargador se nos informará también de este extremo en la pantalla).

A continuación encontraremos el listado del programa cargador. Tecleémoslo cuidadosamente y grabémoslo antes de ponerlo en funcionamiento.

```
9000 REM *CARGADOR HEXADECIMAL*
9010 REM *CARLOS DE LA OSSA VILLACANAS*
9020 DEF FN D(H$)=CODE H$-55+7*(H$<"A")
9030 DEF FN B(H$)=16*FN D(H$(1))+FN D(H$
(2))
9040 LET D$="SENTENCIA DATA No.- "
9050 LET DATA=0: LET CONT=0: LET M=1
9060 INPUT "DIRECCION DE UBICACION  ",DI
R
9070 IF NOT M THEN  PRINT : PRINT  BRIGH
T 1;"COMPLETADA LA CARGA": PRINT : PRINT
 "LONGITUD, ";CONT;" BYTES": STOP
9080 LET SUMA=0: LET DATA=DATA+10
9090 READ A$,TOTAL
9100 FOR I=1 TO LEN A$-1 STEP 3
9110 IF A$(I)=CHR$ 42 THEN  LET M=0: GO
TO 9150
9120 LET H$=A$(I TO I+1)
9130 POKE DIR,FN B(H$): LET DIR=DIR+1: L
ET SUMA=SUMA+FN B(H$): LET CONT=CONT+1
9140 NEXT I
9150 IF SUMA<>TOTAL THEN  PRINT D$;DATA;
CHR$ 32; FLASH 1;"ERRONEA": GO TO 9070
9160 PRINT D$;DATA;CHR$ 32;"CORRECTA": G
O TO 9070
```

Grupo de manipulación de bits.



SALTO DE CABALLO

ESS probable que muchos de nosotros compartamos una afición mutua: el sano deporte mental del «salto de caballo». Pues bien, con este programa podremos generar, a plena voluntad, todos los saltos de caballo que deseemos.

EL PROGRAMA

Este programa se compone de un soporte BASIC muy simple, que sirve para gestionar la toma de datos y para la visualización del resultado generado por la auténtica inteligencia del programa: la rutina en código máquina.

Los datos que ésta emplea para la confección del pasatiempo son los siguientes:

1. «FRASE»: a este INPUT debemos responder con la frase que deseamos utilizar en el salto de caballo, SEPARANDO SUS SILABAS MEDIANTE UN ESPACIO, y teniendo en cuenta que no son admisibles sílabas de más de tres letras.
2. «DIMENSION PRIMERA»: número de filas de que se compone el salto de caballo.
3. «DIMENSION SEGUNDA»: número de columnas de que se compone el pasatiempo.
4. «NUMERO DE SILABAS»: número de sílabas que se utilizan en el salto. Si este número es superior al total de sílabas que integran la frase, el cuadro del pasatiempo será completado con espacios vacíos.
5. «FILA INICIAL»: fila de la matriz elegida para

¡!

Según las características requeridas al pasatiempo, el programa tardará más o menos tiempo en su confección, aunque para aumentar su velocidad, hasta han sido eliminadas las llamadas a la subrutina de interrupción.

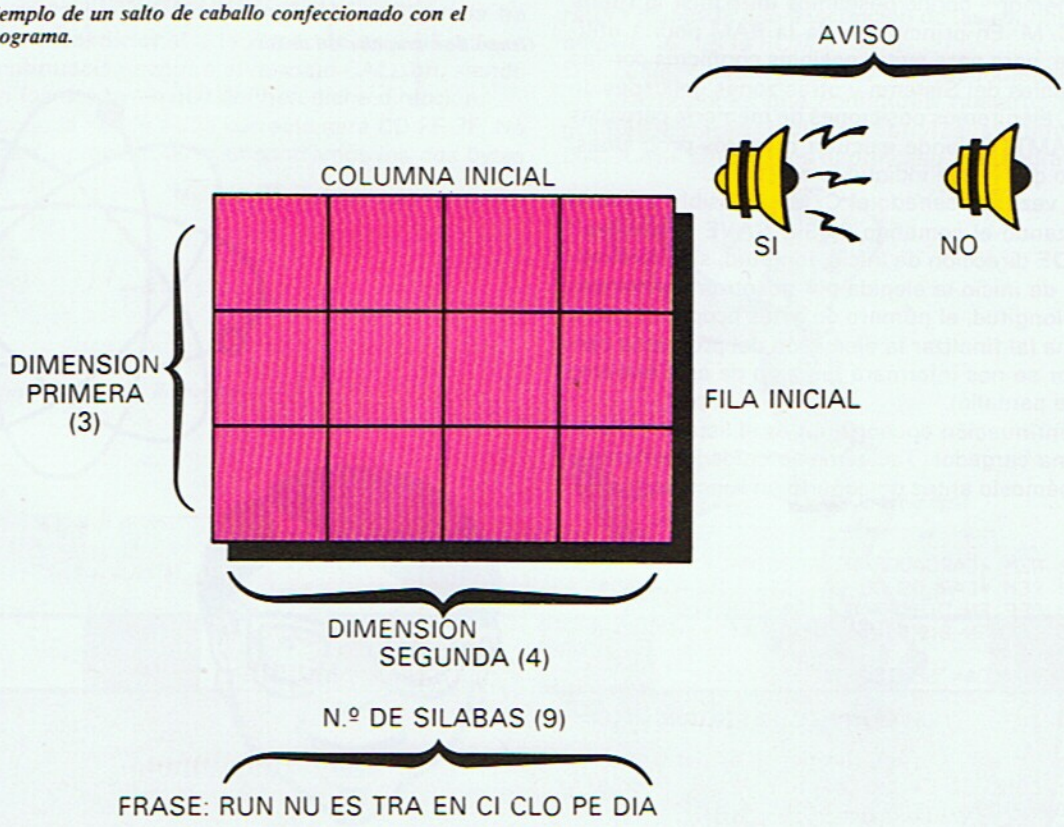
*

En caso de que no sea posible realizar un salto de caballo con las características que indiquemos, el programa nos lo hará saber mediante un mensaje.

*

Para la grabación del programa utilizaremos **SAVE ''S.CABALLO''**, o **SAVE ''S.CABALLO'' LINE 1**, si deseamos activar el sistema de autoejecución.

Ejemplo de un salto de caballo confeccionado con el programa.



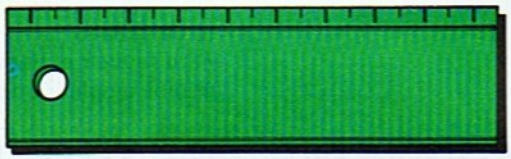
el comienzo del salto. La primera columna es la cero.

6. «COLUMNA INICIAL»: igual que «fila inicial», pero referido a las columnas.

7. «AVISO S/N»: emisión o no de un pitido intermitente como aviso de que ha finalizado la confección del salto de caballo.

La rutina en C/M actúa poniendo en práctica técnicas recursivas, gracias a lo cual su ocupación de espacio es mínima. A partir de la casilla de origen, el programa comprueba todas las series de

RUN ES ME JOR



Las sílabas que integren la frase a tratar no deben sobrepasar en ningún caso los tres caracteres.

movimientos posibles, escogiéndolos de forma aleatoria.

Si al final de la secuencia elegida, no localiza un salto posible, da un paso atrás en su camino y comprueba otra nueva vía.

Pese al sistema de programación empleado, el tiempo de confección de un pasatiempo puede ser grande, si las características que hemos requerido de él así lo exigen. De igual modo, es posible que el programa llegue a la conclusión de la imposibilidad material de realizar el salto de caballo deseado, cuestión ésta que nos anunciará mediante la emisión del mensaje: "NO existe tal salto de caballo".

Para reducir el tiempo de confección del salto podemos emplear un pequeño truco: definir un cuadrado con más casillas que sílabas tiene la frase a tratar. De este modo, se simplificará la labor de distribución por parte del programa, que completará las casillas sobrantes con tres asteriscos (***).

Para grabar el programa, basta con la utilización del siguiente comando: SAVE "S.CABALLO". Si prefiriésemos la opción con autoejecución, teclearíamos la orden de grabación de la siguiente forma: **SAVE "S.CABALLO" LINE 1.**

Para confeccionar un salto de caballo es necesario expresar la frase a tratar con sus sílabas separadas por espacios.

RUN NU ES TRA EN CI CLO PE DIA

```
   1 DATA 221,033,199,129,221,009,221,126,000,201,221
,033,207,129,221,009,221,126,000,201
   2 DATA 120,203,039,203,039,203,039,203,039,177,079
,006,000,221,033,207,130,221,009,201
   3 DATA 237,091,197,129,033,187,129,001,005,000,237
,176,042,197,129,001,005,000,009,034
   4 DATA 197,129,201,042,197,129,001,005,000,055,063
,237,066,034,197,129,017,187,129,001
   5 DATA 005,000,237,176,201,062,255,050,191,129,033
,191,129,052,006,000,033,191,129,078
   6 DATA 205,215,129,033,190,129,134,033,188,129,119
,205,225,129,033,189,129,134,033,187
   7 DATA 129,119,254,000,250,192,130,033,193,129,150
,242,192,130,058,188,129,254,000,250
   8 DATA 192,130,033,192,129,150,242,192,130,237,075
,187,129,205,235,129,221,126,000,214
   9 DATA 000,194,192,130,058,196,129,221,119,000,033
,194,129,190,250,144,130,033,195,129
  10 DATA 054,001,195,192,130,205,255,129,033,196,129
,052,058,188,129,050,190,129,058,187
  11 DATA 129,050,189,129,205,044,130,033,196,129,053
,205,022,130,058,195,129,254,000,194
  12 DATA 192,130,237,075,187,129,205,235,129,221,054
,000,000,058,195,129,254,000,192,058
  13 DATA 191,129,254,007,200,195,049,130
  14 POKE 23658,8: FOR I=33239 TO 33486: READ X: POKE
 I,X: NEXT I
  17 DIM L$(32): DIM A(8,2): DIM B$(256,3)
  20 DATA 2,1,1,2,-1,2,-2,1,-2,-1,-1,-2,1,-2,2,-1
  30 RESTORE 20: FOR I=1 TO 8: READ A(I,1): READ A(I,
2): NEXT I
  40 GO SUB 1000
  50 GO SUB 2000: GO SUB 2525
  55 PRINT AT 3,12; FLASH 1;"ESPERA"
  60 RANDOMIZE USR 33324
  65 IF Q$="S" AND INKEY$="" THEN  BEEP .8,20: PAUSE
10: GO TO 65
  70 GO SUB 1500
  80 INPUT "QUIERES PROBAR OTRA VEZ? ";R$: IF R$<>"N"
 THEN  GO TO 50
  90 GO TO 10000
1000 REM  INPUT FRASE
1010 PAPER 6: BORDER 6: CLS : INPUT "FRASE: ";A$: LET
 A$="*** "+A$+" $ "
1020 LET CONT=1: LET J=1
1030 LET I=J
1040 IF A$(J)=" " THEN  LET J=J+1: LET I=J: GO TO 104
0
1050 IF A$(J)<>" " THEN  LET J=J+1: GO TO 1050
1060 LET C$=A$(I TO J-1)
1070 IF C$="$" THEN  RETURN
1080 LET B$(CONT)=C$: LET CONT=CONT+1
1090 GO TO 1030
1500 REM  SALTO
1505 GO SUB 2500
1510 IF NOT PEEK 33219 THEN  CLS : PRINT AT 10,2;"NO
EXISTE TAL SALTO DE CABALLO": RETURN
1520 LET Y1=INT ((31-N1*3)/2)
1530 LET X1=INT ((21-N2)/2)
1540 FOR I=0 TO N1-1
1550 FOR J=0 TO N2-1
1560 PRINT  INK 9; PAPER 1+6*(((I+J)/2)=INT ((I+J)/2)
);AT X1+J+1,Y1+I*3+1;B$(1+PEEK (33487+16*I+J))
1570 NEXT J: NEXT I
1580 RETURN
2000 REM  INICIALIZACION TABLAS
2010 FOR I=1 TO INT (RND*5)+1
2020 LET J=INT (RND*8)+1: LET K=INT (RND*8)+1
2030 FOR L=1 TO 2: LET X=A(J,L): LET A(J,L)=A(K,L): L
ET A(K,L)=X: NEXT L
2040 NEXT I
2050 FOR I=0 TO 7
2060 POKE 33223+I,A(I+1,1)
2070 POKE 33231+I,A(I+1,2)
2080 NEXT I
2090 FOR I=33487 TO 33743: POKE I,0: NEXT I
2100 RETURN
2500 REM  PANTALLA
2510 BORDER 2: PAPER 5: CLS : PRINT  PAPER 6;AT 0,0;L
$;AT 21,0;L$
2520 FOR I=1 TO 21: PRINT  PAPER 6;AT I,0;" ";AT I,31
;" ": NEXT I: RETURN
2525 REM  ENTRADA DATOS
2526 GO SUB 2500
2530 PRINT AT 6,5;"DIMENSION PRIMERA: ";AT 8,5;"DIMEN
SION SEGUNDA: ";AT 10,5;"NUMERO DE SILABAS: ";AT 12,5
;"FILA INICIAL:";AT 14,5;"COLUMNA INICIAL:";AT 16,5;"
AVISO (S/N):"
3010 INPUT "DIMENSION PRIMERA: ";N2;: PRINT AT 6,24;N
2: POKE 33216,N2: IF N2<=0 OR N2>10 THEN  BEEP .2,20:
 GO TO 3010
3020 INPUT "DIMENSION SEGUNDA: ";N1: PRINT AT 8,24;N1
: POKE 33217,N1: IF N1<=0 OR N1>16 THEN  BEEP .2,40:
GO TO 3020
3030 INPUT "NUMERO DE SILABAS: ";NCUA: PRINT AT 10,24
;NCUA: POKE 33218,NCUA: IF NCUA<=0 OR NCUA>N1*N2 THEN
  BEEP .2,40: GO TO 3030
3040 INPUT "FILA DE LA CASILLA INICIAL: ";Y: PRINT AT
 12,24;Y: POKE 33214,Y: IF Y<0 OR Y>N1 THEN  BEEP .2,
40: GO TO 3040
3050 INPUT "COL. DE LA CASILLA INICIAL: ";X: PRINT AT
 14,24;X: POKE 33213,X: IF X<0 OR X>N2 THEN  BEEP .2,
40: GO TO 3050
3055 INPUT "AVISO? (S/N): ";Q$: PRINT AT 16,24;Q$
3060 POKE 33219,0: POKE 33220,2: POKE 33221,0: POKE 3
3222,144
3065 POKE 33487+X*16+Y,1
3070 INPUT "ESTA BIEN ASI? (S/N): ";R$: IF R$="N" THE
N  GO TO 2526
3080 RETURN
```

BASES DE DATOS

EN general, el concepto de Base de Datos es bastante amplio. De hecho, conocemos comúnmente por este nombre a cualquier sistema informatizado de gestión de datos que facilita el acceso a éstos para consulta o modificación.

Aunque en principio este tipo de gestión masiva de información podemos decir que está reservado a grandes ordenadores (por motivos de capacidad de almacenamiento), no es menos cierto que los microordenadores juegan un papel importante en su explotación. De hecho, en muchos países puede accederse desde un ordenador doméstico a los «Bancos» o Bases de Datos.

Una de las aplicaciones tradicionales de las Bases de Datos es el acceso rápido para consultas de estudiantes o empresarios a informaciones de lo más dispar, las cuales van desde el desarrollo de temas sobre Historia, Literatura, Ciencias Naturales, etc... al del Marketing o la Bolsa.

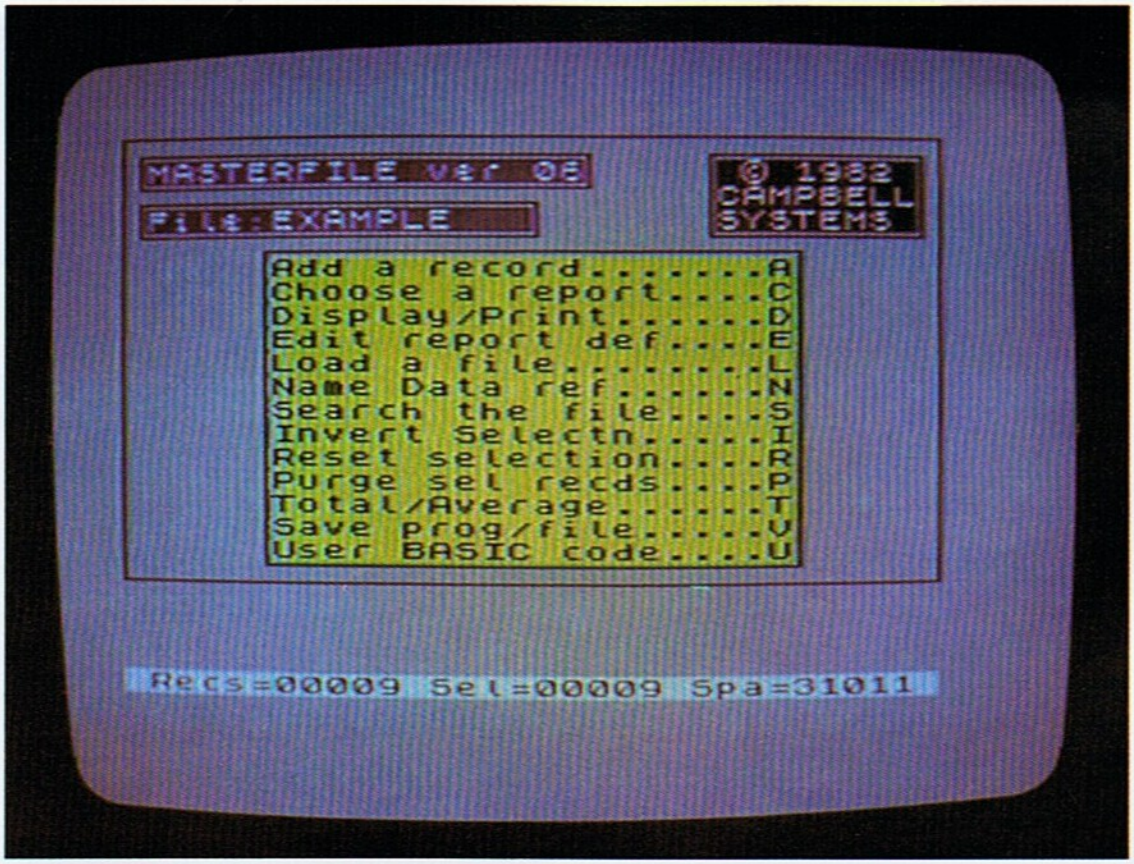
Otra de las características es el modo de tratamiento y acceso a los datos. En las Bases de Datos se permiten tanto accesos al azar *(Random)* como secuenciales, así como la combinación de ambos métodos. Es decir, es posible solicitar información sobre un tema para obtener una primera información y, a partir de ésta, conseguir una ampliación de detalles importante.

Sin embargo, y a pesar que este es el concepto tradicional de lo que es una Base de Datos, existe otra acepción de idéntico nombre, pero con muy distinto desarrollo, del mismo vocablo.

Cuando hablamos de microordenadores entendemos por Base de Datos al programa de utilidad que nos permite el manejo de ficheros indexados, es decir, el acceso aleatorio a la información.

Además, las Bases de Datos de microordenado-

El menú principal del programa MASTERFILE permite el acceso a trece opciones. Este puede ser sobreimpreso en la pantalla en cualquier momento en el modo DISPLAY.



Los datos a introducir en MAILIST son: nombre, título, compañía, dirección 1, dirección 2, ciudad, país (o provincia en su caso), código postal, teléfono y claves de búsqueda.

*

El programa MAILIST es apto para los modelos de 16 ó 48 *Kbytes* permitiendo el mantenimiento de 9 ó 213 direcciones, respectivamente.

*

MAILIST dispone de un conjunto de 16 caracteres por registro para albergar claves que se pueden asignar dependiendo de determinadas circunstancias.

res nos permiten estructurar nuestra información en forma de registros y campos, de manera que pueden obtenerse salidas a pantalla o impresas de aquellos datos que cumplan o no con determinadas condiciones, diferentes modos de clasificación, selección o eliminación de registros etc...

En este sentido, son varios los programas comerciales que convierten nuestro Spectrum en una verdadera «agenda electrónica», un complejo fichero con información sobre clientes o propietarios de un edificio, etc...

Dada la imposibilidad de abarcar el abanico de posibilidades que se nos ofrece en este sentido, nos ocuparemos de algunas de las más características.

MASTERFILE (CAMPBELL SYSTEMS)

MASTERFILE es una completa Base de Datos destinada al modelo de 48 Kbytes, que permite destinar casi 32 Kbytes al almacenamiento estructurado de información. Entre sus características más notables podemos citar:

— Dirigido por Menú.
— Registros de longitud variable.
— Nombres de informes y datos definibles por el usuario.
— Clasificación por cualquier campo de datos.
— Búsqueda de registros con cualquier criterio de selección.
— Salida de 1 a 22 registros por pantalla.
— Impresión comprimida (hasta 51 caracteres por línea).
— Salida a la impresora ZX.
— Salida a impresoras profesionales vía *interface*.
— Acumulación y análisis porcentual.
— Posibilidad de actualización, duplicado y borrado de registros.
— Carga y grabación de ficheros de datos y del propio programa.
— Acceso opcional a rutinas escritas por el usuario.
— Compatible con las unidades MICRODRIVE.
— Alta velocidad de operación.

Los ficheros son manejados como conjuntos de

¡!

El fichero de demostración VU FILE contiene los datos: nombre, continente, capital, idiomas, moneda, población, renta per cápita y extensión.

*

VU FILE es otra de las tentativas, sin duda la más cuidada en cuanto a la presentación de los datos en pantalla, de cubrir la necesidad de mantener ficheros de datos.

*

VU FILE es una creación de PSION apta para los modelos de 16 y 48 *Kbytes* con la que suministran dos ficheros de demostración, conteniendo los países de Europa y del Mundo, respectivamente.

Mediante la denominación inglesa DATA BASE se conocen aquellos programas destinados al soporte de información en ficheros: las BASES DE DATOS.

datos almacenados en casete o cartucho los cuales serán cargados en RAM para consulta o actualización. En cuanto al contenido del fichero, puede ser tan simple como un listín telefónico o tan complicado como un fichero de datos del personal de una empresa. En cualquier caso, un fichero está compuesto por registros, cada uno de los cuales referido a un elemento determinado, una persona, un artículo de un almacén, etc...

Cada registro MASTERFILE puede contener hasta 26 datos diferentes, denominados campos, como pueden ser un nombre, apellido, número de teléfono, o el *stock* de un artículo. Todos los campos de datos son tratados por MASTERFILE como de longitud variable (hasta 128 caracteres). No existe pues una estructura rígida para los registros, por lo que pueden colocarse los diferentes campos de datos en el orden deseado, de cara a su posterior presentación en pantalla o salida a impresora. Los campos son referenciados por medio de una letra, que no debe repetirse en ninguno de los demás.

Una de las particularidades más significativas de MASTERFILE es la posibilidad de clasificar los datos en determinado orden, para su visualización o impresión, sin que por ello se vea afectada la estructura inicial de los datos. Además, puede elegirse también entre diferentes formatos de informes para la presentación de los datos, definibles por el usuario y que pasan a formar parte del fichero al almacenar su contenido sobre soporte magnético.

Existe la posibilidad de asociar palabras a cada campo a modo de título (o nombres de campo), de forma que éstos sean referenciados de esta manera en la petición de los datos, para una mayor facilidad de identificación.

La tarea más importante es decidir qué datos se albergarán en cada registro, ocupando los campos de la A a la Z. Para calcular el tamaño del registro puede usarse la fórmula:

T=1+N+C

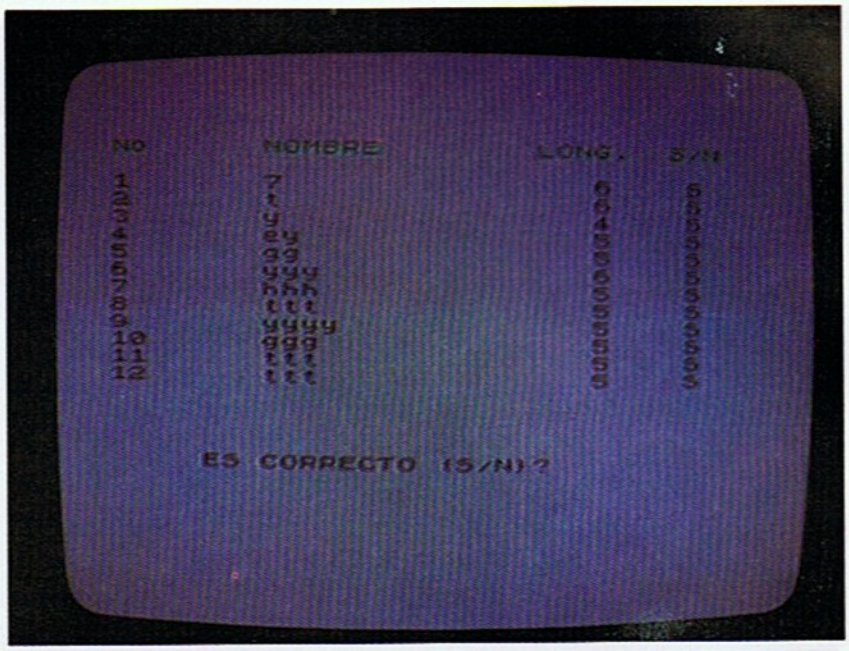
Donde **N** es el número de campos del registro y **C** la longitud de los campos en caracteres, para conseguir al dividir **32000/T** el máximo número de registros posible.

Los menús MASTERFILE admiten como respuesta letras mayúsculas o minúsculas. Cuando el programa se carga en memoria presenta automáticamente el Menú Principal **MM** *(Main Menu)*.

El Menú Principal nos informa también del nombre del fichero (hasta 10 caracteres) y el número de versión del programa.

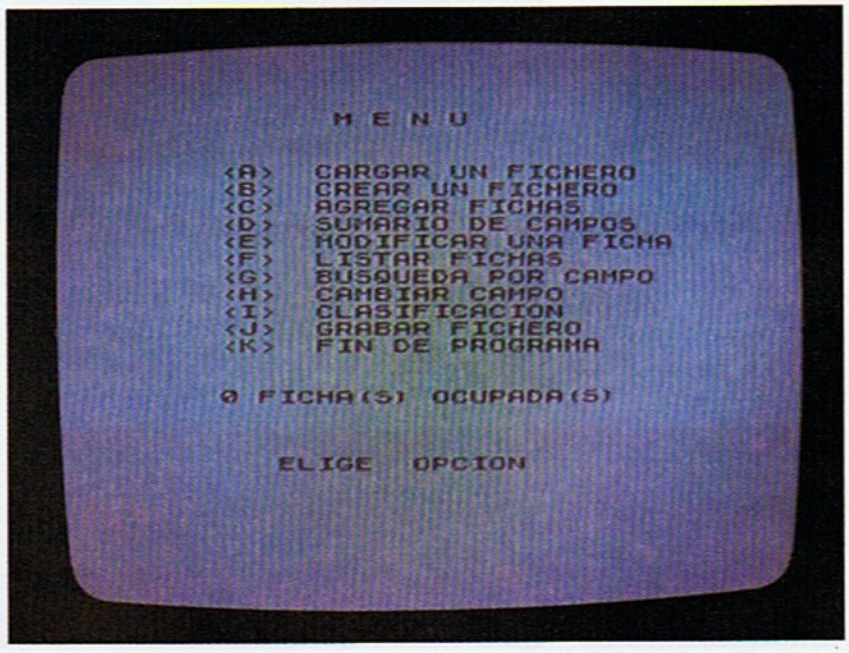
Cuando se responde con la pulsación de una opción concreta del menú, el programa ejecuta la acción indicada o pasa a ofrecer otro menú complementario. Todas las respuestas a los menús se producen por la pulsación de un único carácter; sin embargo, en algunas ocasiones el programa requiere una contestación en forma de texto, lo que indicará por medio del cursor parpadeando en el modo **L**, concluyendo en este caso la introducción por medio de la tecla **ENTER**.

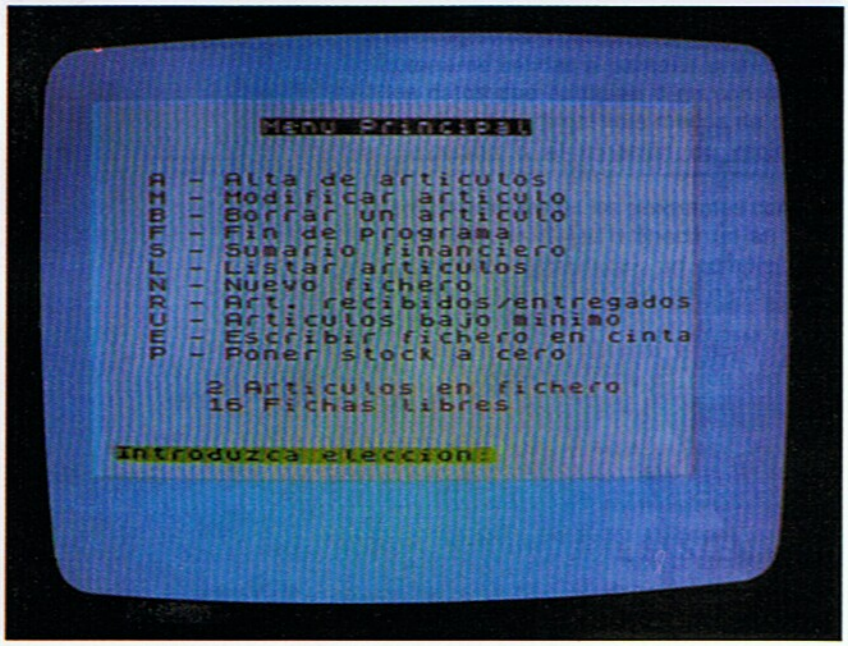
MASTERFILE ofrece casi permanentemente en pantalla un menú o mensaje, sin embargo, existe una excepción: cuando nos encontramos en el



Una opción habitual en las bases de datos es la del listado de las características que definen los diversos campos de un registro del fichero.

Los menús principales son la llave de acceso a todas las opciones que nos presenta el programa en cuestión.





Los programas de gestión de STOCKS se pueden considerar como una extensión especializada de las bases de datos de propósito general.

modo **DISPLAY.** En este caso, podemos sobreimpresionar el menú en pantalla usando la tecla **Q** *(Quit),* pulsándola de nuevo para hacer desaparecer el menú.

El programa opera en gran variedad de modos, generalmente identificados en la cabecera de la página, con excepción del modo **DISPLAY** que reserva las líneas 0 a 21 para el usuario, utilizando la línea 22 para los mensajes del programa.

Podría ser peligroso permitir que la simple pulsación de una tecla destruyera información, por eso el programa siempre pide una confirmación por medio de la tecla **Y**, considerándose cualquier otra respuesta como intención de abandonar la opción.

Las posibilidades de selección de registros son grandes. Pueden seleccionarse los registros que cumplan determinada condición, o efectuar la función inversa, es decir, seleccionar cierto grupo de registros para su eliminación. Además pueden obtenerse totales de los campos de datos numéricos seleccionados en los informes.
La anchura normal de la pantalla es de 32 caracteres. Sin embargo comprimiendo los caracteres a un ancho de 6 ó 5 *bits,* en lugar de la anchura normal de 8, pueden conseguirse dos grados diferentes de aprovechamiento de la línea: 42 ó 51 caracteres.

VU-FILE es seguramente la base de datos de calidad con mayor veteranía en el mercado.

¡!

Todos los campos de datos son tratados por MASTERFILE como de longitud variable (hasta 128 caracteres).

*

BASE DE DATOS no sólo es capaz de gestionar la información que le suministramos, sino que es capaz de generar nuevos campos de datos calculables mediante fórmula a partir de unos datos base.

BASE DE DATOS (ABC SOFT)

Las opciones que VU-FILE dispone para el tratamiento de los ficheros se resumen en un menú presentado simultáneamente con la ficha en la pantalla.

BASE DE DATOS es otra de las opciones para la gestión de ficheros de datos destinada al modelo de 48 *Kbytes,* con la particularidad de reunir además ciertas posibilidades propias de una hoja electrónica. Este programa no sólo es capaz de gestionar la información que le suministramos, sino que es capaz de generar nuevos campos de datos calculables mediante fórmula a partir de unos datos base.

La aplicación está guiada por un menú de 10 opciones que incluyen: creación del fichero, introducción de datos, modificación, supresión, búsquedas, ordenaciones, listados, generación de campos por fórmula, grabación y borrado de ficheros.

El primer paso es la creación del fichero. El programa va pidiendo los nombres de los campos de datos que componen el registro (10 caracteres como máximo), tipo (numérico y alfanumérico) y longitud máxima (20 para numéricos y 100 para alfanuméricos). Pueden definirse hasta 35 campos.

Los campos operacionales permiten hacer cálculos a partir de los campos numéricos definidos como base en la creación del fichero, de forma que al variar los campos de éstos, los campos operacionales recalculan automáticamente su valor, como si de una hoja electrónica se tratara. Pueden crearse hasta 5 campos operacionales que se renumeran a partir de los ya creados. Para crearlos es preciso introducir el nombre, el número de campo sobre el que se desea operar (puede ser también una constante) y, seguidamente, la operación (+, –, * ó /) deseada. Este proceso puede repetirse para un máximo de 3 campos, que se actualizarán según sean modificados los campos base.

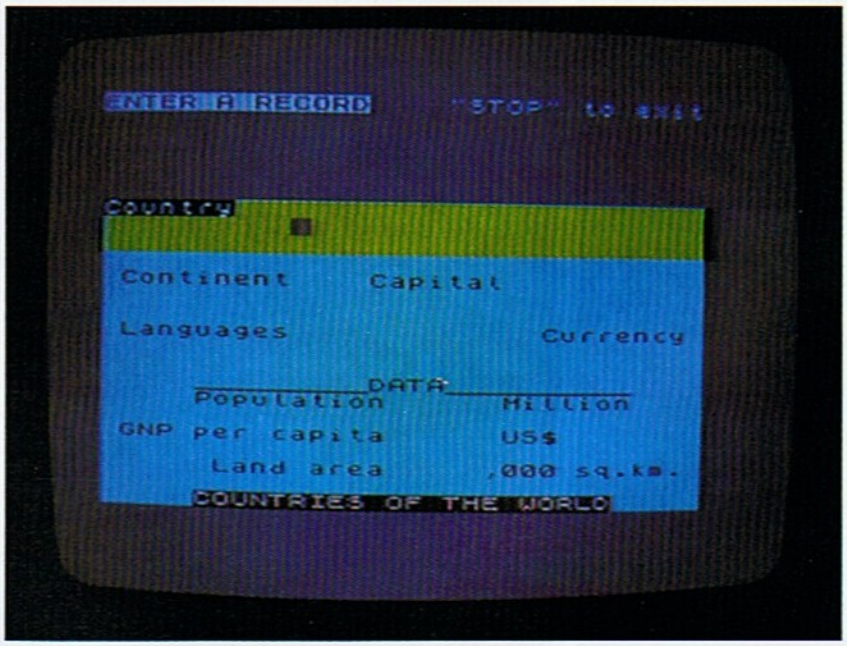
Para la introducción de datos se presentan en distinto tono los nombres de los campos y su tamaño, abandonando este modo al pulsar **ENTER** en la primera posición del primer campo de datos del registro. En cualquier otra posición, la pulsación de **ENTER** sirve de paso al campo siguiente.

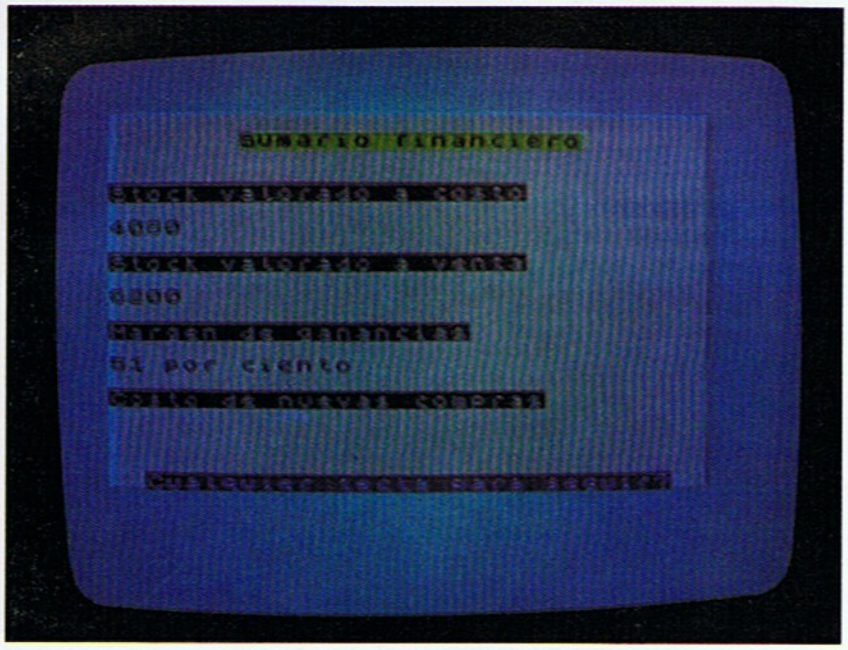
Las teclas de movimiento del cursor nos permiten el acceso a la línea superior o inferior, del mismo modo, **CAPS SHIFT** y **0** permite eliminar caracteres.

Para modificar un registro o ficha se nos pide la clave, permitiendo modificarla de una forma similar a la referida para la entrada de datos. Asimismo, podemos eliminar los registros con sólo indicar su clave.

La búsqueda se verifica seleccionando el campo adecuado (puede ser operacional), tecleando a

En el VU-FILE la grabación de nuevas fichas es de una extraordinaria sencillez, puesto que una ficha en blanco se representa en la pantalla, siendo recorrida por un cursor.





Una base de datos especializada en gestión de STOCKS proporciona determinadas informaciones en base al contenido de las fichas que no ajustaría una aplicación de soporte de ficheros de tipo general.

La capacidad es un dato muy a tener en cuenta en el tipo de programas que tratamos; la utilidad del programa como el de la foto, con una capacidad para sólo nueve fichas, es cuando menos muy discutible.

continuación la serie de caracteres a buscar (20 como máximo). Después de pedir conformidad, el programa nos muestra los registros que cumplen la condición, pudiendo actuar sobre ellos con la forma de edición habitual.
Puede ordenarse el fichero por cualquier campo. Para listarlo, se selecciona entre qué claves y si se quiere impreso. Si así fuera, podemos optar por una impresora serie (indicando la velocidad de transmisión). Nos permite introducir condiciones eludiéndose así el listado de determinadas fichas.

MAILIST (ABC SOFT)

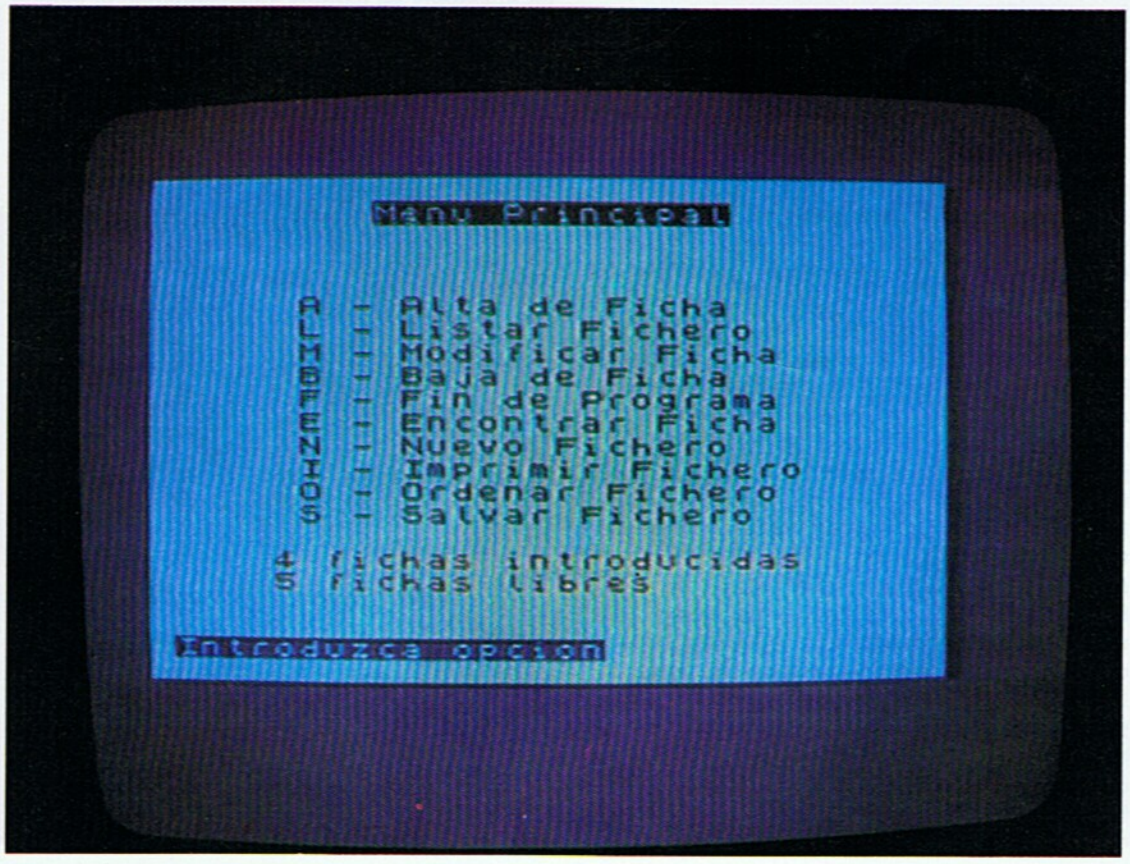
Este programa es apto para los modelos de 16 ó 48 *Kbytes* permitiendo el mantenimiento de 9 ó 213 direcciones, respectivamente. El programa permite el almacenamiento y recuperación de nombres y direcciones, los cuales son almacenados permanentemente en cinta y pueden ser car-

¡!

MASTERFILE permite clasificar los datos en determinado orden, para visualización o impresión, sin que por ello se vea afectada la estructura inicial de los datos.

*****

MASTERFILE permite elegir también entre diferentes formatos de informes para la presentación de los datos, definibles por el usuario y que pasan a formar parte del fichero al almacenar su contenido sobre soporte magnético.



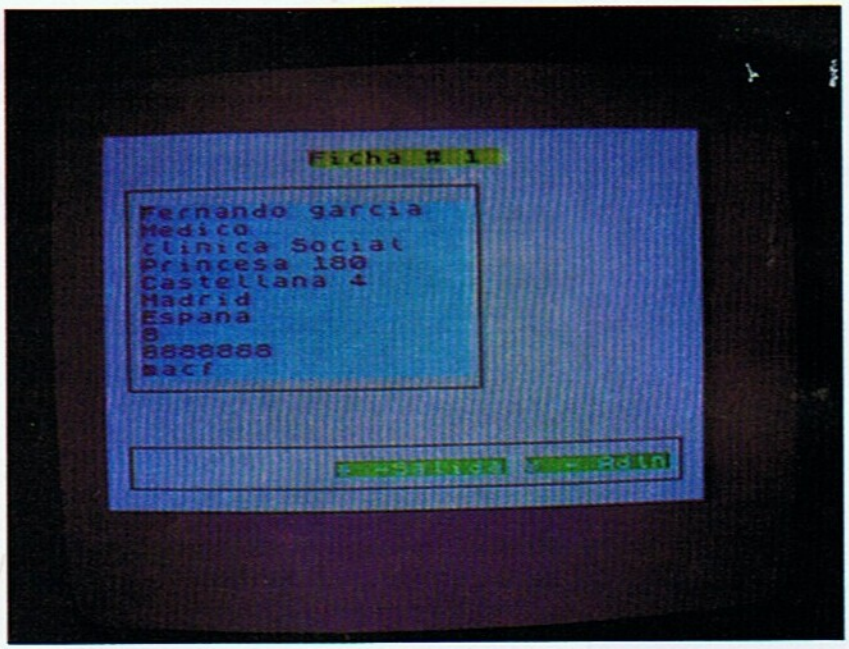
gados en memoria para manejarlos. Dispone, a su vez, de rutinas para añadir nuevos registros, modificar, eliminar y ordenar. También los registros pueden ser visionados o impresos.

La rutina de impresión incluye facilidades para establecer un rango de datos del fichero o para realizar una impresión selectiva. Además dispone de un conjunto de 16 caracteres por registro para albergar claves que se pueden asignar dependiendo de determinadas circunstancias. Cada código es un carácter cualquiera y tiene significado posicional, de forma que facilita una posterior selección de los registros.

Los datos a introducir son: nombre, título, compañía, dirección 1, dirección 2, ciudad, país (o provincia en su caso), código postal, teléfono y claves de búsqueda.

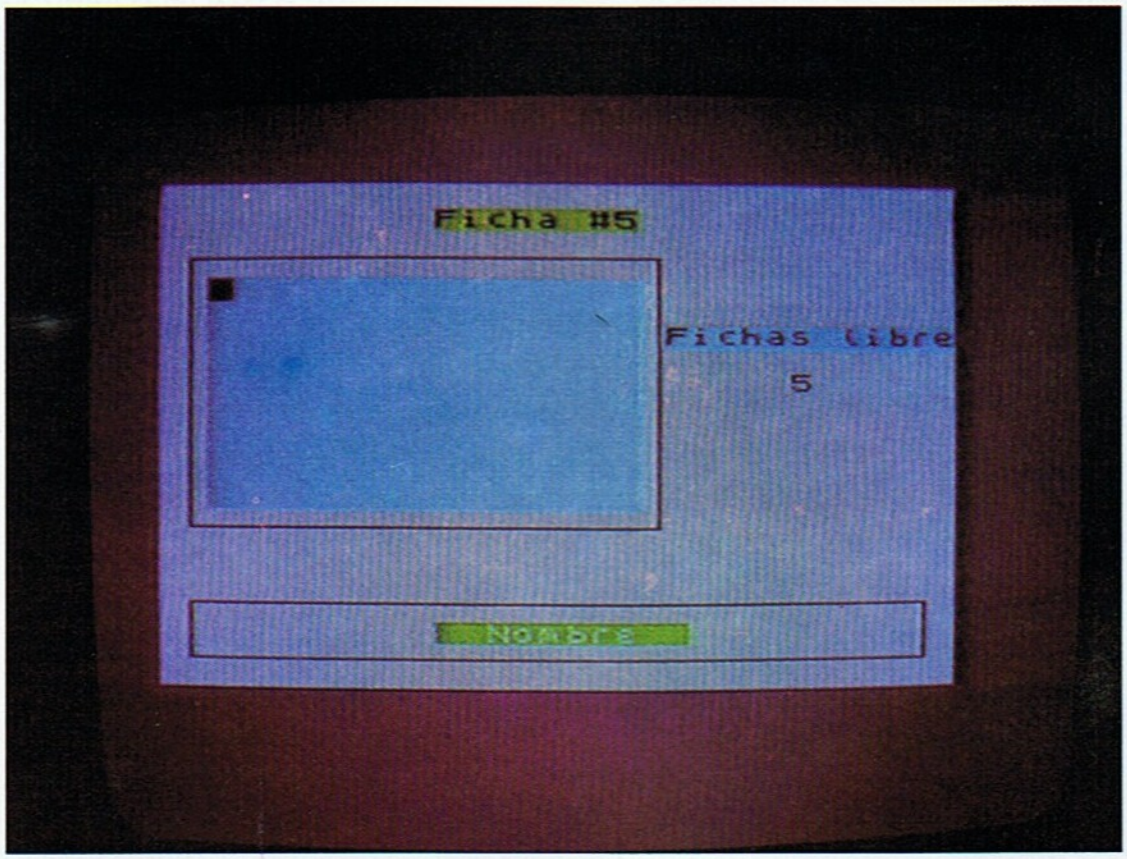
Para seleccionar fichas se elige el campo adecuado y se introduce un valor de búsqueda. El programa muestra todos los registros que cumplen la condición. En el caso del campo 10 (claves de asociación) el proceso se realiza de modo ligeramente diferente: cualquier espacio en blanco es interpretado como deseo de ignorar lo que los registros contienen en dicha posición.

La impresión de etiquetas se produce con un espacio a la izquierda y una línea en blanco por debajo, sin incluir las claves de asociación. Para cambiar estos valores basta con introducir el número de espacios a la izquierda (0 a 9), el espa-



Hemos de procurar que los campos vayan acompañados de una descripción, de lo contrario, el significado de la información presentada puede quedar un poco perdido, tal como sucede en la foto.

Del mismo modo que en la presentación, durante la toma de datos es importante que quede claro el dato que se solicita como sucede en la pantalla de ejemplo.



¡!

MASTERFILE ofrece casi permanentemente en pantalla un menú o mensaje, sin embargo, existe una excepción: cuando nos encontramos en el modo **DISPLAY**.

*

BASE DE DATOS es otra de las opciones para la gestión de ficheros de datos destinada al modelo de 48 *Kbytes*, con la particularidad de reunir además ciertas posibilidades propias de una hoja electrónica.

¡!

Conocemos comúnmente por Base de Datos a cualquier sistema informatizado de gestión de datos que facilita el acceso a éstos para consulta o modificación.

*

En las Bases de Datos se permiten tanto accesos al azar *(Random)* como secuenciales, así como la combinación de ambos métodos.

*

La anchura normal de la pantalla MASTER-FILE es de 32 caracteres. Sin embargo comprimiendo los caracteres a un ancho de 6 ó 5 *bits*, pueden conseguirse dos grados diferentes de aprovechamiento de la línea: 42 ó 51 caracteres.

*

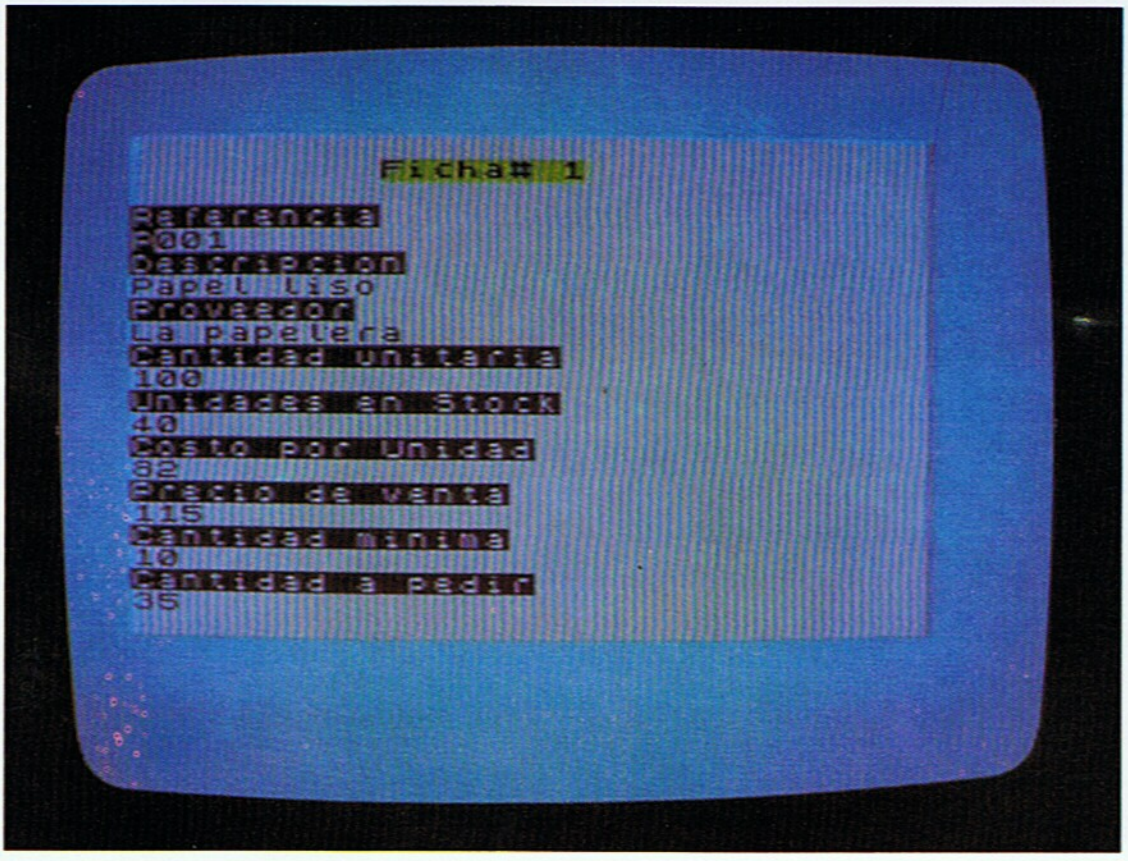
Una de las aplicaciones de las Bases de Datos es el acceso rápido para consultas de estudiantes o empresarios a informaciones que van desde la Historia, Literatura, Ciencias Naturales, etc... al Marketing o la Bolsa.

ciado vertical (0 a 9), y una S si se desea que las claves de asociación figuren en la primera línea de cada etiqueta.
Por último, el programa posee la opción de obtener por impresora un listín telefónico conteniendo los nombres y teléfonos de los registros.

VU FILE (PSION COMPUTERS)

VU FILE es otra de las tentativas, sin duda la más cuidada en cuanto a la presentación de los datos en pantalla, de cubrir la necesidad de mantener ficheros de datos. VU FILE es una creación de PSION apta para los modelos de 16 y 48 *Kbytes* con la que suministran dos ficheros de demostración, conteniendo los países de Europa y del Mundo, respectivamente. El fichero contiene los datos: nombre, continente, capital, idiomas, moneda, población, renta per cápita y extensión.
El programa consta de una zona de texto BASIC que maneja las rutinas de carga y grabación de ficheros y un bloque de código máquina para gestionar la creación y mantenimiento de los registros de datos, así como del formato de presentación en pantalla, el cual puede definirse exactamente a la medida del usuario.
En resumen, el programa permite las siguientes opciones: entrada, modificación, informe sobre el estado del fichero, acceso al próximo registro, acceso al anterior, ordenación por cualquier campo de datos, selección, listado en pantalla, listado en impresora, copia literal de la pantalla en impresora *(hard copy)* y eliminación de registros no necesarios.
Quizá una de las características más importante es la facilidad a la hora de definir la estructura que adoptarán los datos en pantalla. Para ello, disponemos de una zona de 20 filas por 32 columnas. Dentro de esta zona de pantalla podemos posicionarnos en cualquier punto para determinar las cabeceras de los datos. Una vez concluida la operación, se pasa a definir el comienzo de cada campo de datos, pudiendo elegir tanto el color del **PAPER** como el de la **INK**. De forma similar, puede definirse también el formato que tendrá la salida de los datos a la impresora.

La denominación de campo para aplicaciones concretas es también muy especial; en la foto, gracias a estas denominaciones, adivinamos de inmediato que nos encontramos ante una gestión de STOCKS.



DIRECCIONANDO

CON el presente capítulo completamos la exposición de los conceptos previos necesarios para la correcta comprensión del lenguaje máquina, antes de adentrarnos en el estudio detallado de cada instrucción en particular.
Serán tratados temas fundamentales, como la representación de un número en complemento a dos, el cálculo de operaciones aritméticas utilizando este formato, y los diferentes modos de direccionamiento implementados en el microprocesador de nuestro Spectrum.
A partir de este momento, ya es posible considerar que la introducción quedó atrás, y por tanto, es esencial que prestemos una atención especial a todo lo que sigue, pues de su correcta compresión depende en buen grado la operatividad de los programas que construyamos.

EL COMPLEMENTO A DOS

Hasta ahora sabemos que toda información almacenada en la memoria de nuestro ordenador, sea cual sea su naturaleza, se encuentra en forma de dígitos binarios o bits. Como el Z80 es un microprocesador de 8 bits, en cada posición de memoria tendremos ocho de estos dígitos.
Los números siguen, por supuesto, el mismo formato, y sabemos que con un octeto es posible definir 2↑8=256 cantidades distintas comprendidas entre 0 y 255, es decir, entre 00000000 y 11111111 en binario.
Pero obviamente, nuestro ordenador es capaz de manejar números negativos (si no, menudo desastre). Entonces, ¿cómo los reconoce? La respuesta la proporciona una técnica de codificación de números enteros (positivos y negativos) parecida a la binaria que todos conocemos, denominada complemento a dos.
Si un número está codificado según esta técnica, el microprocesador sólo tiene que «ojear» el bit D7 (el de mayor peso) del byte correspondiente para identificar su signo. Es decir, si es 0, el número será positivo, mientras que si encuentra un 1, lo considerará negativo.
Posiblemente estaremos pensando que sólo nos quedan 7 bits con los que efectuar combinaciones y, por tanto, representar con ellos un número como el 255 puede resultar problemático.
Efectivamente, esto es así y para no llevarnos a engaño vamos a definir los márgenes sobre los cuales es posible trabajar con 8 bits en complemento a dos. Despreciando el de signo, podremos

Cálculo del complemento a dos de un número.

89 (decimal)	Ø	1	Ø	1	1	Ø	Ø	1	
	1	Ø	1	Ø	Ø	1	1	Ø	PASO 1
+	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	PASO 2
	1	Ø	1	Ø	Ø	1	1	1	COMPLEMENTO A DOS DE

¡!

Cuando un número está codificado en complemento a dos, el bit D7 es el que indica su signo (0+, 1–).

*

Cuando la suma de dos números positivos en complemento a dos da como resultado uno negativo o a la inversa, se ha producido el denominado desbordamiento o sobrepasamiento.

formar 2↑7=128 combinaciones diferentes, entre 0000000 y 1111111.
Es decir, si a los 7 bits anteriores les añadimos en primer lugar un 0 (indicativo del signo más), los números positivos en este formato estarán comprendidos entre 0 y 127 expresados en decimal. ¿Y los negativos?

CALCULANDO EL COMPLEMENTO A DOS

Para hallar el complemento a dos de un número representado en binario seguiremos los siguientes pasos:
1) Cambiamos sus dígitos 1 a 0, y los que estén a 0 los colocamos a 1.
2) Al número resultante le sumamos 1 expresado en binario, es decir, 00000001 (ver la tabla correspondiente a la suma en la aritmética binaria).
Pongámoslo en práctica con algunos ejemplos. Calculemos el complemento a dos del número 89, 01011001 en binario. Al efectuar el paso primero obtenemos 10100110. A continuación le sumamos 00000001:

DECIMAL	COMPLEMENTO A DOS
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

Representación en complemento a dos de los números con cuatro bits.

¡!

La introducción quedó atrás, siendo esencial que prestemos especial atención a los nuevos conceptos vertidos en el presente capítulo.

*

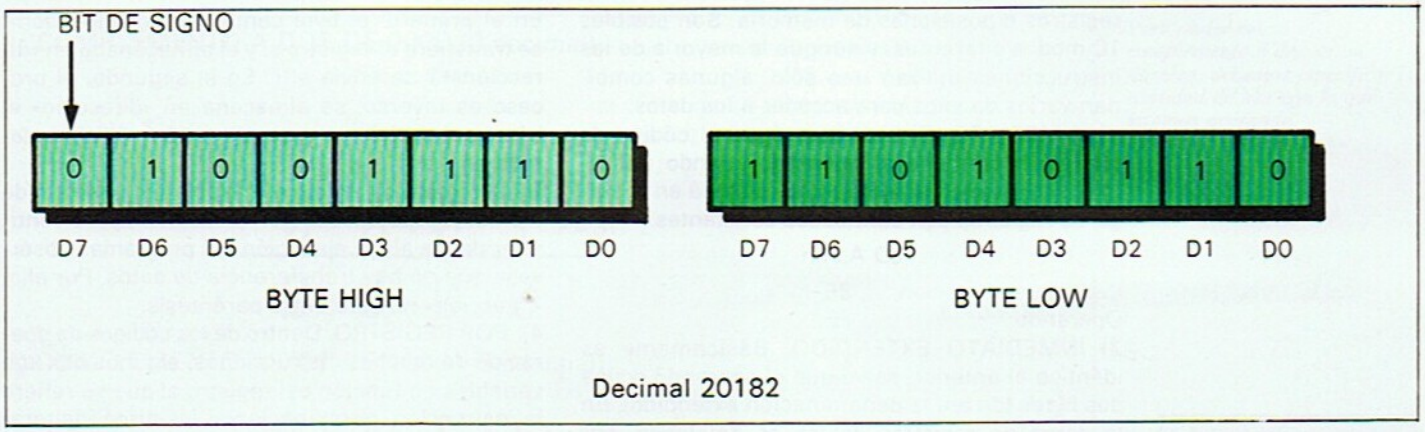
Toda la información almacenada en la memoria de nuestro ordenador está en forma de dígitos binarios o bits.

La suma en la aritmética binaria.

+	1	Ø
1	Ø	1
Ø	1	Ø

```
 10100110
+00000001
 10100111 = -39 decimal
            (observa que el bit D7 es 1)
```

Del presente ejemplo, y de los que muestran las figuras (ejercitémonos con ellos y comprobemos que llegamos a los mismos resultados), podemos extraer las siguientes importantes consecuencias:
— Siempre debemos especificar el número de bits, cuando trabajamos en complemento a dos, con el que estamos operando.
— El complemento a dos de un número positivo es siempre negativo, y viceversa.
— Utilizando 8 bits es posible representar números enteros entre −128 y +127. En general, si operamos con n bits, el complemento a dos permite cantidades entre −(2↑(n−1)) y +(2↑(n−1))−1.
— El complemento a dos de un número decimal negativo es su código binario.
— Cuando utilizamos números de 16 bits en complemento a dos, el bit D7 del byte de mayor peso contiene el signo de éste, siguiendo el mismo criterio anterior.
— Dos números en complemento a dos «se complementan», es decir, si los sumamos obtenemos



Signo en los números de 16 bits en complemento a dos.

cero (recordemos que ha de estar claro el número de bits con el que estamos trabajando).

— Para sumar dos números en complemento a dos se siguen las mismas reglas que si estuvieran en su codificación binaria.

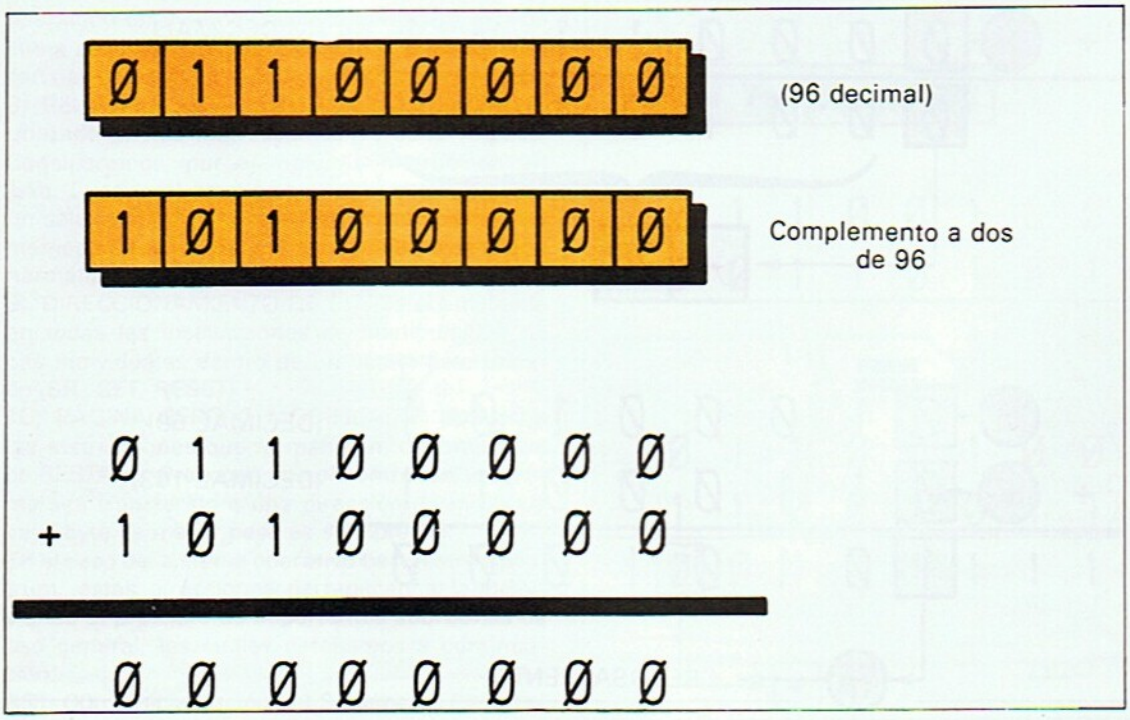
— La resta se efectúa complementando el sustraendo y sumándolo al minuendo.

— Cuando la suma de dos números positivos da uno negativo, o la de dos negativos es positiva, se ha producido el denominado sobrepasamiento o desbordamiento. En tal caso, podremos tener constancia de dicha situación analizando el contenido del bit D2 del registro F, el cual es colocado a 1 si sucediera esta circunstancia.

MODOS DE DIRECCIONAMIENTO

Con direccionamiento queremos hacer referencia a la forma en que el microprocesador de nuestro Spectrum accede los datos almacenados en los

Dos números en complemento a dos «se complementan».



registros o posiciones de memoria. Son posibles 10 modos diferentes, y aunque la mayoría de las instrucciones utilizan uno sólo, algunas combinan varios de ellos para acceder a los datos.

1) INMEDIATO. El byte que sigue al código de operación contiene al operando cuando utilizamos este modo. Por ejemplo, se emplea en la carga de registros con cantidades constantes:

LD A,FFh

Código de operación: 3E
Operando: FF

2) INMEDIATO EXTENDIDO. Básicamente es idéntico al anterior, salvo que el operando ocupa dos bytes (de ahí la denominación extendido). En la carga de registros dobles es empleado, por ejemplo, este modo:

LD HL, FFA9h

Código de operación: 21
Operandos: A9 (byte LO) y FF (byte HI)

3) EXTENDIDO. Las instrucciones que lo manejan contienen siempre en sus dos últimos bytes una dirección de memoria. Esta puede ser el lugar donde recoger o almacenar el dato o a donde ha de saltar el programa para seguir ejecutándose. Genéricamente, podría representarse del siguiente modo:

LD rs,(dirección)
LD (dirección),rs
JP dirección

En el primero, el byte contenido en «dirección», se transfiere al registro s, y el almacenado en «dirección»+1 se envía al r. En el segundo, el proceso es inverso: se almacena en «dirección» el byte contenido en s y en «dirección»+1, el contenido en r.

En el tercero se produce un salto a la posición de memoria especificada por «dirección», para continuar desde allí la ejecución del programa. Obsérvese que no hay transferencia de datos. Por ello, «dirección» no está entre paréntesis.

4) POR REGISTRO. Dentro de los códigos de operación de muchas instrucciones, algunos bits son variables en función del registro al que se refiere la instrucción, permaneciendo los otros inalterables. Tal es el caso de la transferencia de datos entre registros, por ejemplo:

LD A,B (cargar en el acumulador el byte contenido en el registro B)

5) INDIRECTO. Su forma de operar es similar al direccionamiento extendido, sólo que en este caso se emplea un par de registros para señalar a la posición de memoria de donde recoger el dato. Por ejemplo:

LD A,(HL)

que significa: «cargar en A el byte contenido en la posición de memoria a la que apunta el registro doble HL».

6) IMPLICITO. Las instrucciones que lo manejan conocen a priori la posición de los datos implica-

¡!

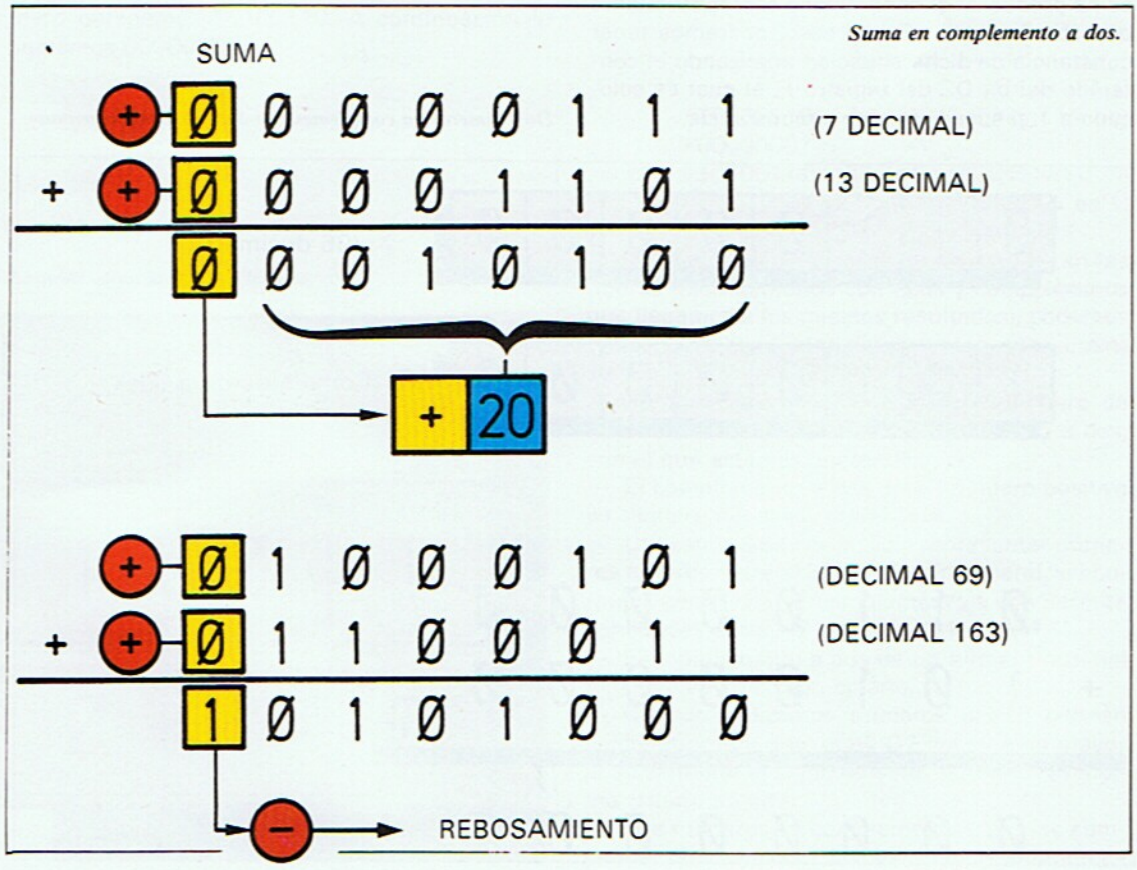
Cuando en el mnemónico que representa una instrucción encontramos un registro entre paréntesis, significa «el contenido de la dirección señalada por el registro».

*

Las instrucciones de RESTART provocan una llamada a ciertas direcciones de la ROM de nuestro Spectrum.

*

El bit D2 del registro F nos avisa si ha habido desbordamiento en una operación aritmética.



Suma en complemento a dos.

1) COMPLEMENTO A DOS DE 0000 (0 decimal)

```
   1 1 1 1   PASO 1
+  Ø Ø Ø 1   PASO 2
-----------
   Ø Ø Ø Ø
```

2) COMPLEMENTO A DOS DE 1000 (-8 decimal)

```
   Ø 1 1 1
+  Ø Ø Ø 1
-----------
  [1] Ø Ø Ø
   ↓
¿NEGATIVO?
```

EL COMPLEMENTO A DOS DE UN NUMERO NEGATIVO HA DE SER POSITIVO

-8 NO TIENE COMPLEMENTO A DOS UTILIZANDO 4 BITS

Para poder calcular el complemento a dos de un número, debemos especificar la cantidad de bits con la que estamos operando.

dos. Este es el caso de las operaciones aritméticas y lógicas de 8 bits, porque todas ellas realizan sus operaciones con el acumulador, por ejemplo:

ADD A,B

la cual suma los contenidos de los registros A y B, dejando el resultado en A.

7) INDEXADO. La mayoría de las instrucciones que manejan los registros índice IX e IY lo implementan. Todas ellas, contienen tras el código de operación un byte de desplazamiento (d), que se suma al contenido actual del registro índice para señalar a una determinada posición de memoria.

8) RELATIVO. Es el modo de direccionamiento utilizado en las instrucciones de salto relativo. Consideremos, por ejemplo, la instrucción JR 1Ah. Cuando el programa la encuentra, efectúa un salto incondicional relativo a una posición de memoria situada 26 bytes más adelante de la que normalmente se habría ejecutado a continuación.

9) DIRECCIONAMIENTO DE BIT. Es el empleado en todas las instrucciones de manipulación de bits individuales dentro de un octeto determinado (BIT, SET, RESET).

10) PAGINA CERO O MODIFICADO. Son ocho las instrucciones que lo manejan, denominadas de RESTART. Provocan que el control del programa sea transferido a una dirección de memoria cuyo byte de mayor peso es siempre 00.

En el caso del sistema operativo de nuestro Spectrum, estas direcciones pertenecen a la ROM, siendo el principio de importantes subrutinas de uso general, las cuales detallamos a continuación:

RST 00h: inicialización del Sistema.
RST 08h: salida de error.
RST 10h: imprimir un carácter.
RST 18h: leer un carácter.
RST 20h: leer el siguiente carácter.
RST 28h: entrada a las rutinas del calculador.
RST 30h: crear espacio en la zona de trabajo.
RST 38h: chequeo del teclado. Subrutina de interrupción.

RESTA

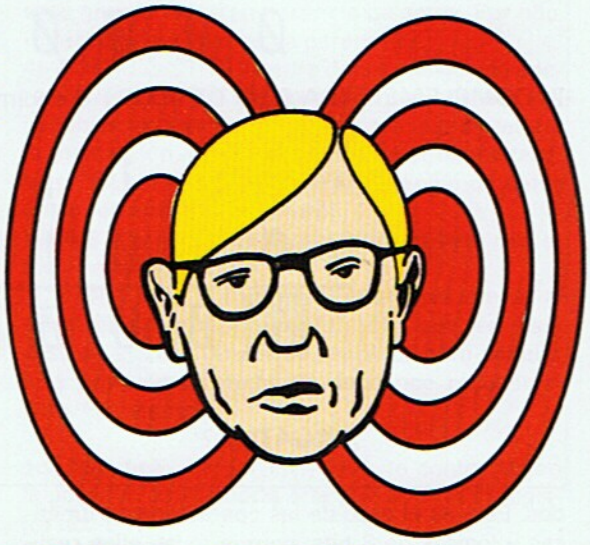
X + Y = X + (-Y)

```
   Ø 1 1 Ø Ø 1 Ø Ø   (DECIMAL 100)
-  Ø Ø 1 1 Ø 1 Ø 1   (DECIMAL 53)
-----------------
        ═
   Ø 1 1 Ø Ø 1 Ø Ø
+  1 1 Ø Ø 1 Ø 1 1   COMPLEMENTO A DOS DE
-----------------
  [Ø] Ø 1 Ø 1 1 1 1
   → [+] 47
```

Resta en complemento a dos.

NIM

CON este programa, nuestro Spectrum dejará de ser nuestro fiel amigo, convirtiéndose en un adversario implacable. Con NIM, vuelve a hacerse presente el eterno duelo, humano-máquina.

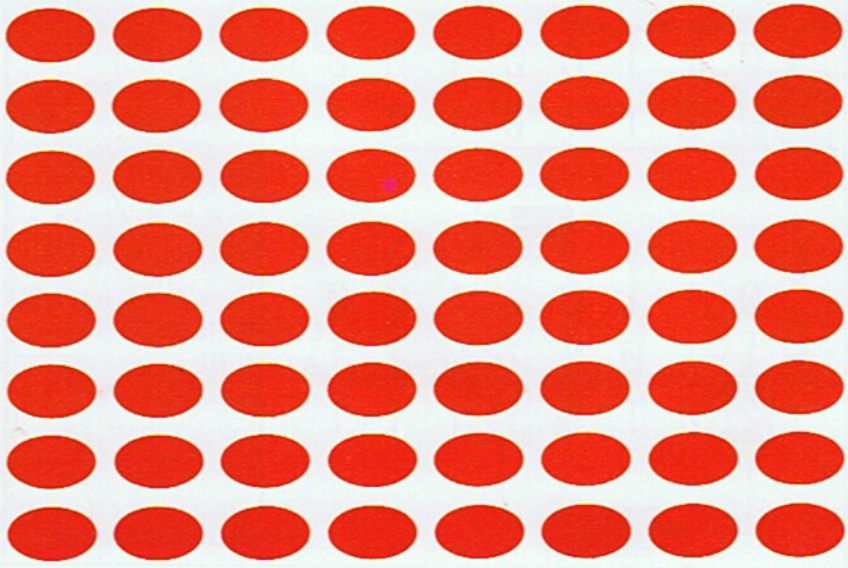


Con NIM vuelve a cobrar vida el eterno duelo humano-máquina.

EL PROGRAMA

El programa nos permitirá hacer trabajar a nuestro cerebro a pleno rendimiento contra nuestro Spectrum, haciendo de él un temible e implacable enemigo.
El programa muestra en la pantalla entre tres y ocho columnas, seleccionables por nosotros, con una cantidad variable de piezas cada una. El número de elementos que componen cada una de estas columnas, es elegido aleatoriamente por nuestro Spectrum, y oscila entre 1 y 8.
Por cada jugada se puede retirar el número de piezas que se desee, pero siempre de la misma columna, siendo obligatorio retirar al menos una.
Gana el jugador que consigue llevarse en su tirada la última pieza, independientemente del número de elementos que haya en dicha jugada.
La obtención del movimiento por parte del ordenador, se basa fundamentalmente en dos subrutinas: una de decisión de jugada óptima y otra aleatoria. Atendiendo al grado de dificultad se efectúa un RND en la línea 600, de forma que se produce una bifurcación en el programa hacia la rutina aleatoria u óptima. Así, en caso de escoger el nivel de dificultad 0, siempre se efectuará un movimiento del tipo aleatorio, y en el caso del nivel 5, siempre del tipo óptimo.
En cualquiera de los otros grados del nivel de dificultad, cuanto menor sea éste, mayor será la probabilidad de que la máquina efectúe un movimiento erróneo.

El programa muestra en la pantalla entre 3 y 8 columnas seleccionables por nosotros, con una cantidad de piezas que varía aleatoriamente entre 1 y 8.



PROGRAMADA PARA VENCER

Lo cierto es que en su nivel máximo de juego, el Spectrum se convierte prácticamente en un recalcitrante enemigo invencible, puesto que ha sido programado con un algoritmo matemático que le indica en todo momento la jugada óptima que debe realizar.

Por la propia esencia de este juego, una vez que uno de los jugadores ha adoptado una posición de «privilegio» en el tablero, el otro no puede hacérsela perder, a no ser, lógicamente, que éste cometa algún error. Gracias a la capacidad de cálculo del ordenador, éste nunca comete ese «error», sino que más bien se apresura a tomar inmediatamente la situación de privilegio (en el nivel máximo), lo cual le convierte en un enemigo implacable.

Siendo un poco observadores, llegaremos a la conclusión de que dejando dos piezas, cada una en una columna, tendremos asegurada la victoria; lo mismo ocurrirá si conseguimos dejar cuatro piezas distribuidas por igual en dos columnas, y muchas otras situaciones. Ahora bien, seleccio-

nar de entrada, y por el método de «a simple vista», una configuración ideal al comienzo del juego es algo bien difícil.

De hecho, el Spectrum no opera de un modo tan «humano» sino que profundizando un poco en el programa, podremos observar que para la selección del movimiento óptimo, el ordenador determina en primer lugar, la forma binaria del número de piezas que hay en cada columna. Una vez hecho esto, busca la paridad o imparidad binaria de cada posición de bit, y la global de todas las columnas: la posición de privilegio del juego consiste en establecer y conservar una paridad binaria par.

```
  10 REM J.M.MAYORAL & F.LOPEZ MARTINEZ
  20 BORDER 2
  21 PAPER 2
  23 INK 9
  25 POKE 23609,100
  26 POKE 23658,8
  30 CLS
  40 PRINT AT 21,0;"NUMERO DE COLUMNAS? (3-8) "
  50 INPUT L
  60 IF L<3 OR L>8 THEN  GO TO 50
  70 PRINT AT 21,0;"NIVEL DE DIFICULTAD (0-5)"
  80 INPUT D
  90 IF D<0 OR D>5 THEN  GO TO 80
 100 DIM C(L)
 110 DIM B(L,4)
 120 DIM S(4)
 130 LET B$="                                    "
 140 FOR M=1 TO L
 150 LET X=INT (RND*8)+1
 160 LET C(M)=X
 170 NEXT M
 180 CLS
 190 PRINT  PAPER 7; INK 1;"                N I M
     "
 200 PRINT
 210 PRINT "COLUMNA:  ";
 220 FOR C=1 TO L
 230 PRINT  INVERSE 1;C; INVERSE 0;"  ";
 240 NEXT C
 250 PRINT AT 13,0;"CANTIDAD: ";
 260 FOR M=1 TO L
 270 PRINT C(M);"  ";
 280 NEXT M
 290 FOR C=1 TO L
 300 FOR M=1 TO C(C)
 310 PRINT  INK 6; BRIGHT 1;;AT 12-M,C*3+7;"3"
 320 NEXT M
 330 NEXT C
 340 LET J=0
 350 PRINT AT 15,0; PAPER 7;"
     "
 360 PRINT AT 17,0;"MOVIMIENTO:"
 370 PRINT AT 19,0;"PIEZAS:"
 380 PRINT AT 19,23;"NIVEL: ";D
 390 GO SUB 1170
 400 IF RND>.5 THEN  GO TO 570
 410 GO SUB 1140
 420 PRINT  PAPER 3; BRIGHT 1; INK 6;AT 17,23;"TU JUE
GAS"
 430 PRINT AT 21,0;B$
 440 PRINT AT 21,0;"COLUMNA? ";
 450 INPUT M
 460 IF M<1 OR M>L THEN  BEEP 1,-10: GO TO 430
 470 PRINT M,;
 480 PRINT "CANTIDAD? ";
 490 INPUT N
 500 PRINT N
 510 IF N>C(M) OR N<1 THEN  BEEP 1,-10: GO TO 430
 520 GO SUB 1230
 530 GO SUB 1170
 540 IF P THEN  GO TO 570
 550 PRINT  PAPER 3; INK 9; FLASH 1;AT 21,0;"    ENHO
RABUENA,   TU GANAS      "
 560 GO TO 1090
 570 GO SUB 1140
 580 PRINT  PAPER 7; INK 4;AT 17,23;"JUEGO YO "
 590 PRINT AT 21,0;B$
 600 IF RND*10>=D*2 THEN  GO TO 810
 610 FOR M=1 TO L
 620 LET X=C(M)
 630 FOR C=4 TO 1 STEP -1
 640 LET Z=INT (X/2)
 650 LET B(M,C)=X-2*Z
 660 LET X=Z
 670 NEXT C
 680 NEXT M
 690 FOR C=1 TO 4
 700 LET X=0
 710 FOR I=1 TO L
 720 LET X=X+B(I,C)
 730 NEXT I
 740 LET S(C)=X-2*INT (X/2)
 750 NEXT C
 760 LET X=0
 770 FOR I=1 TO 4
 780 LET X=X+S(I)
 790 NEXT I
 800 IF X THEN  GO TO 860
 810 FOR M=1 TO L
 820 IF NOT C(M) THEN  GO TO 850
 830 LET N=INT (C(M)*RND+1)
 840 GO TO 1010
 850 NEXT M
 860 FOR C=1 TO 4
 870 IF S(C) THEN  GO TO 890
 880 NEXT C
 890 FOR M=1 TO L
 900 IF B(M,C)=1 THEN  GO TO 920
 910 NEXT M
 920 LET N=0
 930 FOR C=C TO 4
 940 IF NOT S(C) THEN  GO TO 1000
 950 LET X=2^(4-C)
 960 IF NOT B(M,C) THEN  GO TO 990
 970 LET N=N+X
 980 GO TO 1000
 990 LET N=N-X
1000 NEXT C
1010 PRINT AT 21,0;"COLUMNA? ";M,"CANTIDAD? ";N
1020 PAUSE 100
1030 BEEP .1,40
1040 GO SUB 1230
1050 GO SUB 1170
1060 IF P THEN  GO TO 410
1070 PRINT  PAPER 7; INK 9;AT 21,0;"     LO SIENTO,
  YO GANO      "
1080  FOR I=30 TO 0 STEP -1
1081 BEEP .025,I
1082 NEXT I
1083 GO TO 1100
1090 FOR I=0 TO 30
1091 BEEP .025,I
1092 NEXT I
1100 PRINT AT 7,2;"DESEAS INTENTARLO DE NUEVO?"
1110 IF INKEY$="S" THEN  RUN
1120 IF INKEY$<>"N" THEN  GO TO 1110
1130 GO TO 10000
1140 LET J=J+1
1150 PRINT AT 17,12;J
1160 RETURN
1170 LET P=0
1180 FOR M=1 TO L
1190 LET P=P+C(M)
1200 NEXT M
1210 PRINT AT 19,8;P;" "
1220 RETURN
1230 LET X=M*3+7
1240 FOR I=1 TO N
1250 PRINT AT 12-C(M),X;" "
1260 LET C(M)=C(M)-1
1270 NEXT I
1280 PRINT AT 13,X;C(M)
1290 RETURN
```

HOJAS ELECTRONICAS

Las «hojas electrónicas» son la solución informática de muchos de los problemas que se plantean en gestión. De hecho, podemos decir que no hay trabajo de cálculo que no pueda simplificarse enormemente con su ayuda. El cálculo de una nómina, una factura de venta o la confección de un presupuesto, son ejemplos claros de aplicación de este sistema.

Básicamente una hoja electrónica es una matriz de determinado número de filas y columnas que permite, en la mayoría de los casos, asignar un título extenso para cada fila y otro más reducido a las columnas.

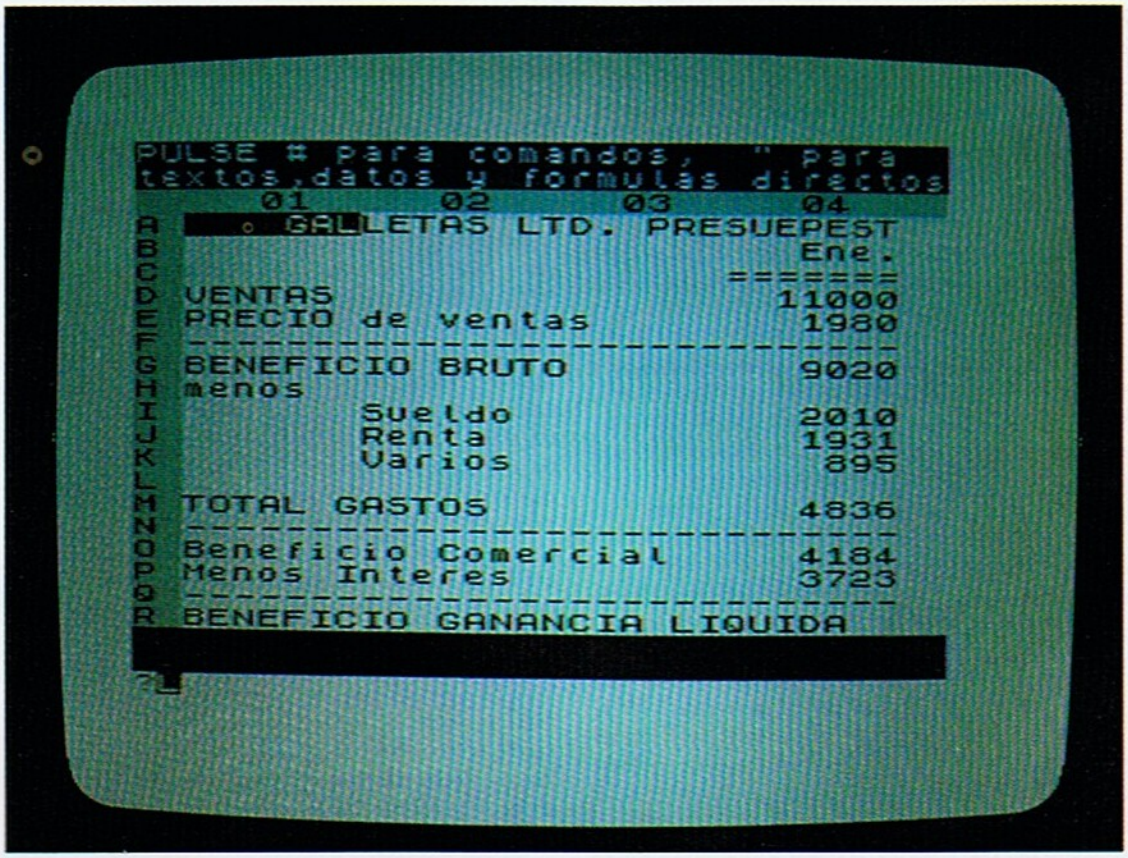
Cada uno de los elementos de la matriz (casilla o celda) es referenciado por su fila y columna, correspondiendo generalmente caracteres alfabéticos a las filas y numéricos a las columnas.

Dentro de estas celdas pueden albergarse tres tipos de datos: constantes alfanuméricas, numéricas y variables calculadas a partir de los contenidos de otras celdas empleando determinada fórmula.

Es el empleo de fórmulas lo que permite que, alterando los valores de determinadas celdas, pueda obtenerse de forma automática el nuevo contenido de otras. Estas fórmulas pueden relacionar cualquier conjunto de celdas por medio de operadores matemáticos, generalmente los elementales (+, –, * y /), aunque algunas hojas electrónicas permiten el empleo de cualquier función BASIC convencional.

En cuanto al número de filas y columnas, así como el ancho de estas últimas y su capacidad de almacenamiento, poco se puede decir de for-

Las hojas electrónicas admiten la entrada tanto de textos como de fórmulas, y por supuesto de los datos básicos para las fórmulas.



¡!

VU-CALC es un programa apto para los modelos de 16 y 48 Kbytes RAM resultando una poderosísima herramienta en análisis financieros, presupuestos, cálculos de ingeniería o de tablas científicas, análisis estadísticos, etc...

*

Otra característica de las hojas de cálculo es la posibilidad de obtener subtotales y totales, tanto de filas como de columnas.

*

La base de datos de MINI OFFICE permite el mantenimiento de un fichero de datos constituido por cierto número de registros (fichas), cuya información se encuentra subdividida en campos (datos elementales).

¡!

El procesador de textos de MINI OFFICE muestra el tiempo transcurrido desde el comienzo de la sesión, el número de palabras escrito y el de caracteres que restan antes que la memoria esté completa.

*

El empleo de fórmulas permite la obtención de forma automática del nuevo contenido de otras.

ma general, puesto que cada programa está dotado de características específicas.
Por último, diremos que otra de las características común a las hojas de cálculo suele ser la posibilidad de obtener subtotales y totales, tanto de filas como de columnas; aunque esta opción no es más que el caso restringido de la aplicación de una fórmula.

VU-CALC (PSION LTD.)

VU-CALC es un programa apto para los modelos de 16 y 48 Kbytes RAM resultando una poderosísima herramienta en análisis financieros, presupuestos, cálculos de ingenieria o de tablas científicas, análisis estadísticos, etc...
La pantalla VU-CALC está dividida en tres zonas: dos líneas de comando en la parte superior, un área en blanco en el centro y una línea para entrada de datos en la parte inferior. La zona central es una «ventana» de la tabla.
Las filas están representadas alfabéticamente y las que se encuentran en la ventana actual, pueden verse a lo largo del margen izquierdo de la pantalla. Las columnas se identifican por números sucesivos, comenzando por 01, y las que pueden verse en la pantalla actual se encuentran justo encima de la ventana de representación de la tabla.

La hoja electrónica puede venir en nuestra ayuda para la resolución de multitud de problemas de cálculo y administrativos.

Los problemas de administración son sin duda muy frecuentes.

Un rectángulo rojo denominado cursor identifica en todo momento la celda en que se está trabajando. Guiados por este cursor pueden leerse datos e introducir textos, fórmulas o cantidades, haciendo uso de las teclas normales de desplazamiento del cursor (flechas de la fila superior del teclado). Al rebasar el movimiento del cursor los límites físicos de la ventana, ésta se desplaza horizontal o verticalmente, en la dirección indicada, para mostrar nuevas zonas de la tabla, ocultas hasta este momento.
Para introducir un texto debe posicionarse el cursor sobre la celda o casilla donde se desea situarlo para, a continuación, pulsar comillas e introducir el texto en cuestión, finalizando la entrada con la pulsación de **ENTER**. La introducción de un valor numérico sigue las mismas normas que el caso anterior, no debiendo incluir las comillas.
La obtención de un resultado en determinada casilla implica la introducción de una fórmula, relacionando cualquier conjunto de casillas. Esta fórmula puede ser comprobada observando la línea de entrada de datos, y si resulta correcta, validarse definitivamente pulsando **ENTER**. Automáticamente, el resultado calculado pasa a ocupar la celda elegida.
Una fórmula puede ser aplicada en diferentes casillas por medio del comando REPETIR. Por medio también de esta tecla, se accede al modo co-

mando, apareciendo los disponibles en la zona superior de la pantalla, seleccionándose por la inicial.

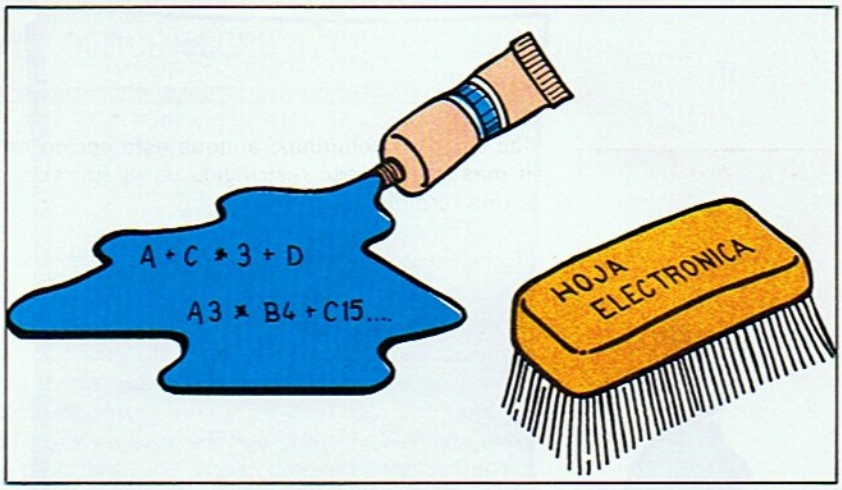
La sintaxis de una fórmula permite incluir constantes, referencias a otras casillas y operadores aritméticos simples: +, –, * y /. Una fórmula puede aplicarse a una celda en concreto o bien repetirse a lo largo de una fila o columna, o por todo un conjunto de ellas.

Las fórmulas se aplican siempre de una forma relativa. Si afecta a una secuencia de celdas de una fila, la columna de referencia se va incrementando siempre, automáticamente, a medida que se aplique la fórmula a lo largo de la fila. Así mismo, el concepto de relatividad es aplicable a columnas o bloques en los que las letras correspondientes a las filas se incrementarán secuencialmente.

En caso de que se desee designar de forma absoluta una celda de referencia en una fórmula, y para que ésta no varíe con el comando repetir, se debe anteponer el prefijo **$** a la celda en cuestión en el momento de escribir la fórmula.

Existe una gran variedad de comandos a los que podemos acceder anteponiendo el símbolo # como primer carácter, seguido de la inicial del comando elegido.

 # B vacía el contenido de la celda indicada por el cursor. **# C** recalcula los valores de todas las celdas, en base a las fórmulas correspondientes. **# E** permite sustituir una fórmula por otra nueva. **# F,c,f,j** especifica el formato de representación de los números en una columna de acuerdo con los parámetros **c, f** y **j**; el primero indica la columna donde aplicar el formato (**A** cuando el formato deba afectar a todas las columnas), el segundo el tipo de formato (enteros o reales con dos decimales) y el tercero indica si desea la justificación a la derecha (**R**) o izquierda (**L**).



Las hojas de cálculo admiten fórmulas con las cuales definir nuestro problema.

 # G,r,c permite desplazar el cursor a la celda indicada a continuación. **# L** limpia la pantalla y carga el fichero que se especifique. **# P** efectúa el *hard copy* de la pantalla en impresora. **# Q** borra el contenido de la hoja dando opción de finalizar el programa.

 # R,rc,f:l provoca la repetición de la fórmula contenida en la celda **rc** a lo largo del rango indicado. **f** y **l** indican las celdas de referencia, donde **f** corresponde a la celda situada más a la izquierda y arriba dentro del bloque y **l** es la situada más abajo y a la derecha; principio y final del bloque, respectivamente.

 # S limpia la pantalla y almacena el contenido de la tabla en casete. **# T,r** o **c,r** o **c** desplaza o copia el contenido de una fila o columna especificada por el primer parámetro a la fila o columna indicada por el segundo, no permitiéndose referir filas a columnas ni viceversa.

El sumatorio es semejante a una fórmula, permitiendo acumular automáticamente parte de una fila, columna o conjunto de celdas. Para ello basta con posicionarse en la celda en que se desea aparezca el resultado de la suma, introduciendo la fórmula con el formato: **& f:l** pulsando **ENTER**; donde **f** y **l** indican la primera y última celda del bloque, respectivamente.

Si el programa se ve interrumpido por causa de un error en los datos, puede continuarse la ejecución tecleando **GO TO 9000** y **ENTER**.

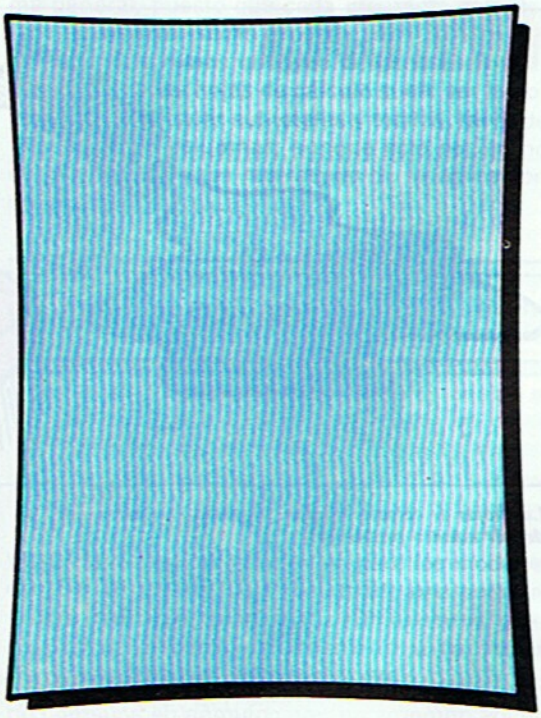
La pantalla es una ventana a la hoja de cálculo.

¡!

MINI OFFICE está preparado para su uso con la impresora ZX PRINTER, o bien con *interfaces* de impresora que no ocupen memoria RAM, como la impresora SEIKOSHA GP50.

*

La pantalla VU-CALC está dividida en tres zonas: dos líneas de comando, un área en blanco en el centro y una línea para entrada de datos en la parte inferior.



Las dimensiones máximas de una hoja de cálculo suelen ser bastante superiores a las de cualquier hoja normal.

OMNI CALC

OMNI CALC es otra de las posibilidades que se presentan para el manejo de información en forma de tabla. Su característica más relevante es la de poder definir, dentro de la lógica limitación de la cantidad de memoria RAM disponible, el número de filas y columnas que se desean.

Este hecho redunda en un mejor aprovechamiento de la memoria, ya que en determinadas aplicaciones puede mantenerse un gran número de filas a costa de reducir el de columnas, como es el caso del mantenimiento de una ficha de cuenta corriente con datos de fecha, movimiento, Debe, Haber y Saldo.

Al cargar el programa en memoria se ofrece la opción de definir una nueva «hoja de trabajo», o de leer una ya creada almacenada en casete. En el primer caso, el programa solicita número de filas y columnas, efectuando la verificación de que el formato indicado tiene cabida en la memoria del ordenador, volviendo a la petición de los datos en caso contrario. Por otro lado, en el caso de seleccionarse la lectura de un fichero ya existente, el programa pide el nombre de éste y lo carga en memoria.

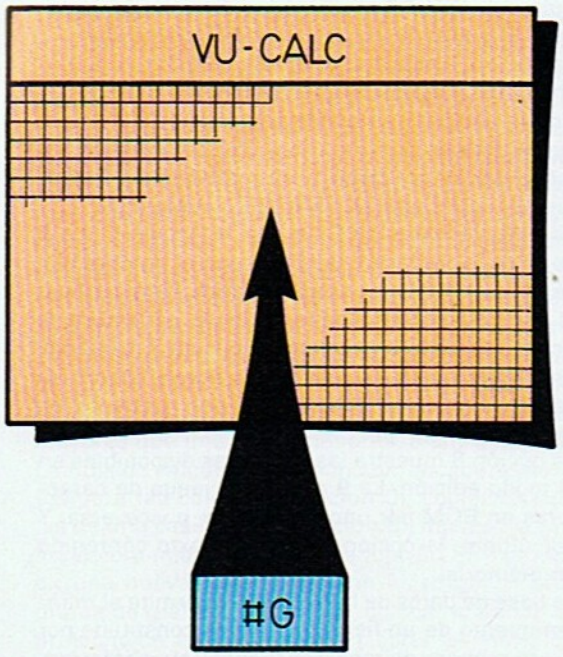
Como es norma general en las «hojas electrónicas» se tiene acceso permanente a una ventana por la cual podemos desplazar un cursor por la pulsación de las teclas **5** a **8**. Con el uso de las mismas teclas en combinación con **CAPS SHIFT**, se consigue el desplazamiento de la ventana a lo largo y ancho de la tabla.

En cada una de las pantallas, además de la porción de datos que podemos ver (bloque de casillas), aparecen permanentemente unos títulos asociados a cada una de las filas, así como los correspondientes títulos de cada columna que se muestra en pantalla.

En cualquier momento, pulsando **H** *(Help)* aparece en pantalla una lista de los comandos que se pueden emplear, los cuales son invocados por su inicial.

C *(Calculate)* recalcula la hoja. **D** *(Decode)* muestra la fórmula asociada a una celda. **G** *(Goto)* desplaza el cursor a la celda indicada. **I** *(Input)* permite introducir textos, valores y fórmulas. **L** *(Load/init)* carga o inicializa la tabla. **M** *(Mode)* permite seleccionar entre la representación de valores en formato entero o dos decimales. **P** *(Print)* copia el contenido de la pantalla o la tabla entera en impresora. **R** *(Repeat)* generaliza una fórmula para un conjunto de celdas. **S** *(Save)* graba en casete el contenido de la tabla bajo el nombre que se indique. **T** *(Total)* calcula subtotales o totales de cualquier fila o columna.

Por ejemplo, en la confección de una nómina, el programa de hoja electrónica puede realizar desde su cálculo hasta su impresión.



El comando #G del programa VU CALC nos permite posicionarnos directamente en cualquier celdilla de la hoja.

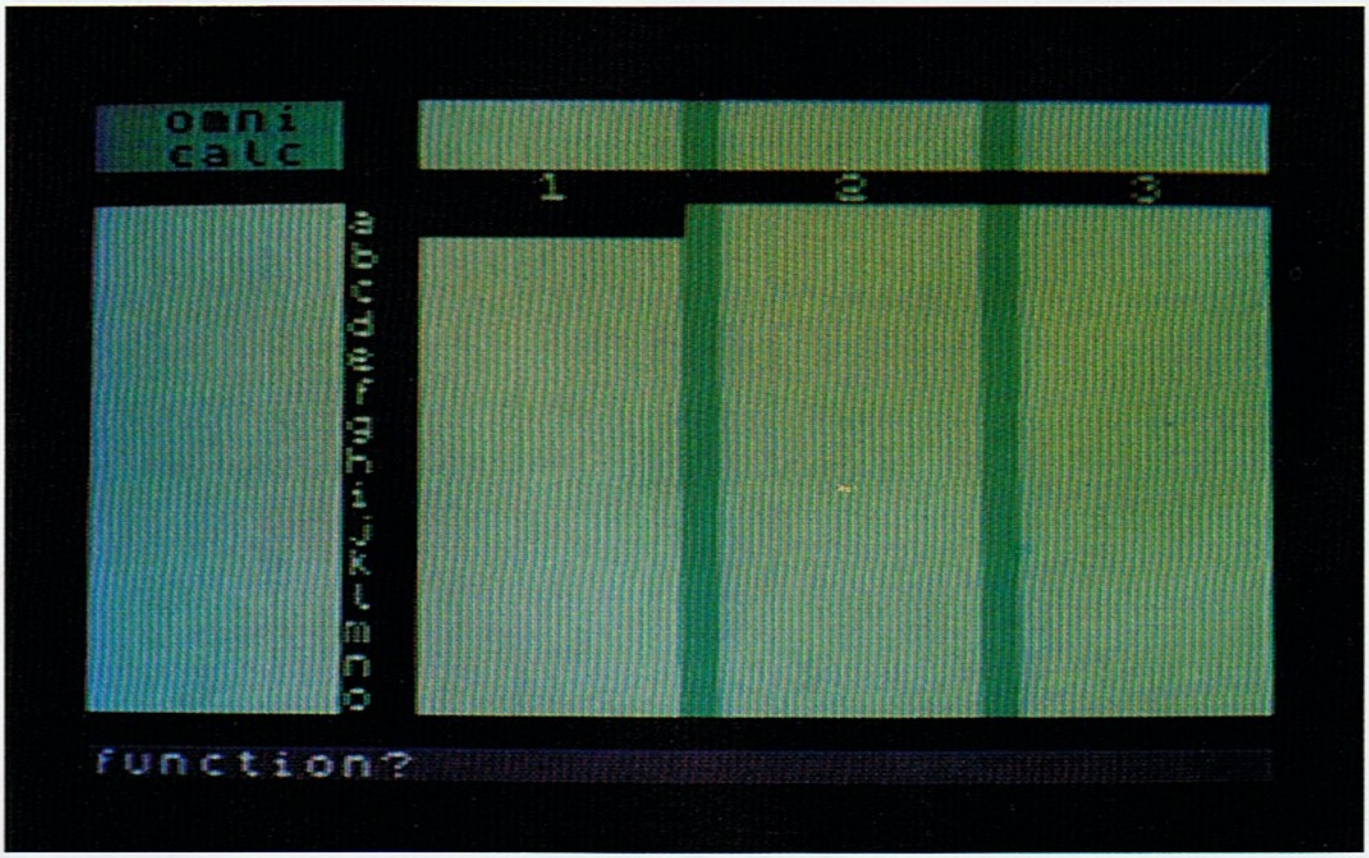
La casilla en que nos encontramos viene señalada por un CURSOR; una franja en el caso de OMNICALC.

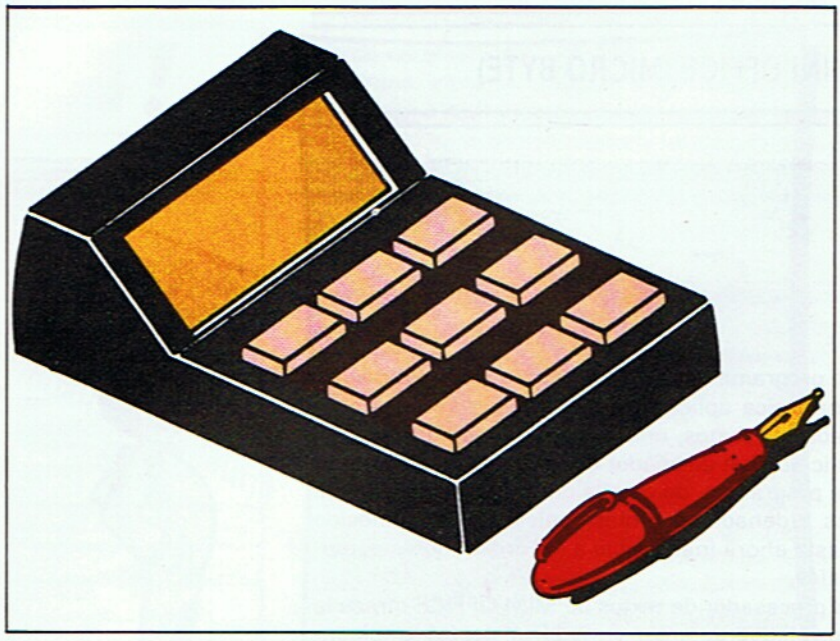
MINI OFFICE (MICRO BYTE)

El programa MINI OFFICE trata de aglutinar en una única aplicación el desarrollo de un procesador de textos, una base de datos, una hoja de cálculo y un diseñador de gráficos. Este conjunto de programas, de amplia difusión en el campo de los ordenadores profesionales, ha permanecido hasta ahora inaccesible a los ordenadores personales.

El procesador de textos de MINI OFFICE ofrece la posibilidad de crear y modificar textos empleando 16 ó 32 caracteres por línea, consiguiendo una gran calidad. En la parte superior de la pantalla se muestra el tiempo transcurrido desde el comienzo de la sesión, el número de palabras escrito hasta el momento y el número de caracteres que restan antes que la memoria esté completa.

Al comenzar con un nuevo texto aparece en su





La hoja electrónica nos evita el engorroso trabajo a base de calculadora y lápiz.

Pulsando **BREAK** en el modo edición se tiene acceso al menú de 10 opciones del procesador.
La opción 1 activa el juego de caracteres de ancho doble y la opción 2 el de tamaño normal.
Con la opción 3 se puede seleccionar el número de caracteres por línea (5 a 32), así como el tamaño de éstos, tanto horizontal como verticalmente, sobre una ZX PRINTER o compatible.
Las tabulaciones se fijan con la opción 4. La 5 permite la grabación del texto con el nombre que se indique (10 caracteres máximo). Para verificar la grabación se emplea la opción 6. La lectura de fichero se lleva a cabo con la opción 7, teniendo en cuenta que la lectura de un nuevo fichero en casete implica la destrucción del residente en memoria.
La opción 8 muestra las funciones disponibles en el modo edición. La 9 cambia el juego de caracteres en ROM por uno alternativo, o viceversa. Y por último, la opción 0 borra el texto contenido en memoria.
La base de datos de MINI OFFICE permite el mantenimiento de un fichero de datos constituido por cierto número de registros (fichas), cuya información se encuentra subdividida en campos (datos elementales) numéricos o alfanuméricos, permitiendo la clasificación del fichero por cualquiera de sus campos.
La primera información que se suministra al programa es el número de campos de que se compone el registro. El siguiente paso es dar nombre a cada uno de los campos especificando al mismo tiempo la clase de datos que albergarán (numéricos o alfanuméricos), así como el número máximo de caracteres por campo, indicando además, en los numéricos, el número de cifras decimales.
Una vez definido el formato del fichero pueden comenzarse a introducir datos (opción A), listar registros por pantalla o impresora (opción F), efectuar búsquedas de determinado dato en un campo (opción G), reemplazar contenidos de campos por uno especificado (opción H), ordenación por uno o varios campos (opción I), suma de valores de un campo (opción D) o grabación del fichero (opción J).

cabecera las palabras COMIENZO y FIN, y entre estos dos límites se va produciendo la entrada y edición del texto.
Por medio de las teclas de control del cursor, se puede acceder a cualquier punto del texto para su modificación o inserción, pudiendo hacer uso de la tecla **DELETE** para eliminar caracteres.

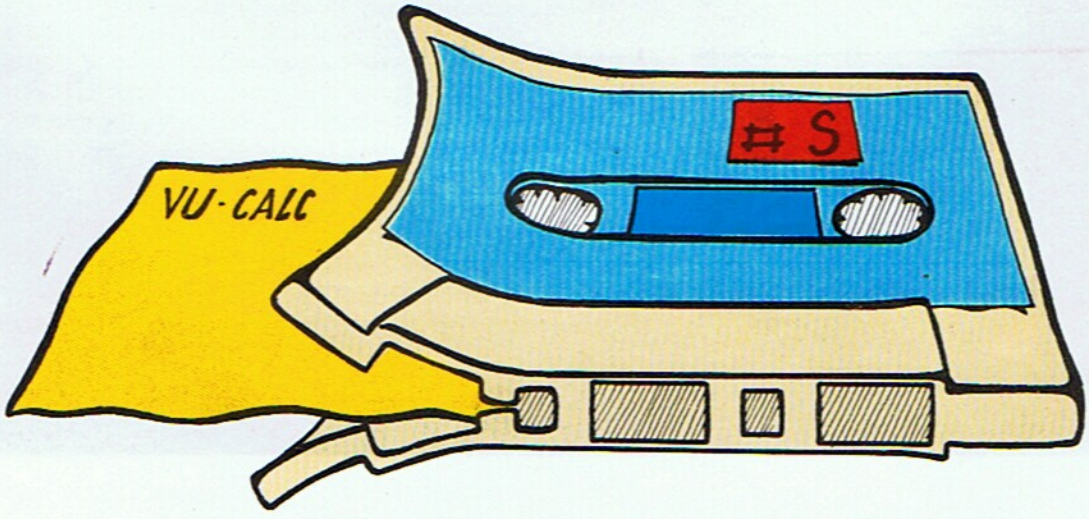
¡!

Las «hojas electrónicas» son la solución informática de muchos de los problemas que se plantean en gestión.

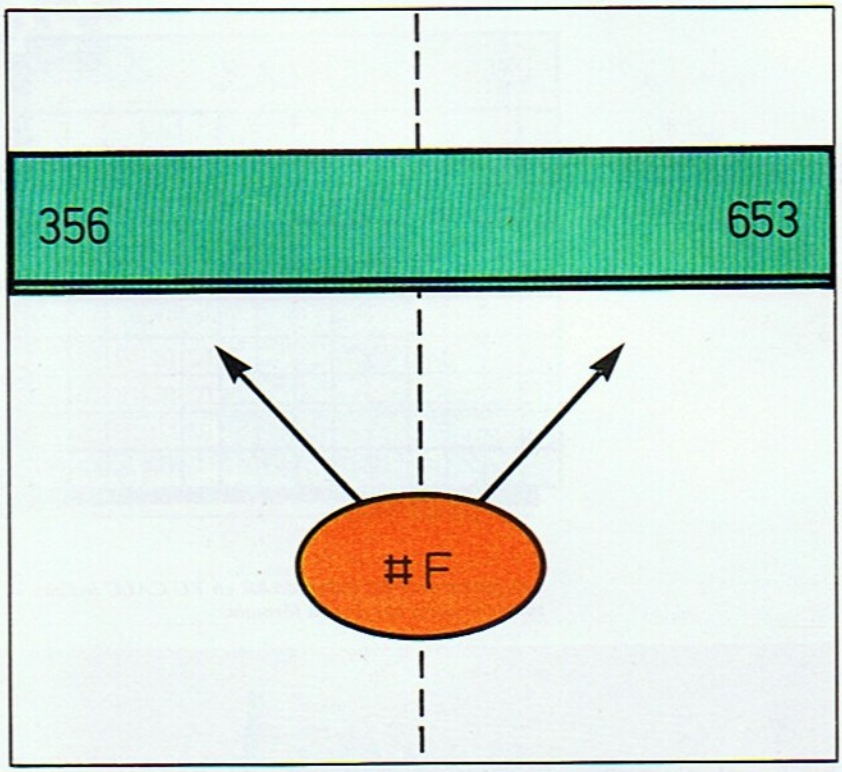
*

El cálculo de una nómina, una factura de venta o la confección de un presupuesto, son ejemplos claros de aplicación de las «hojas electrónicas».

El comando #S almacena el contenido de la hoja VU CALC en casete.

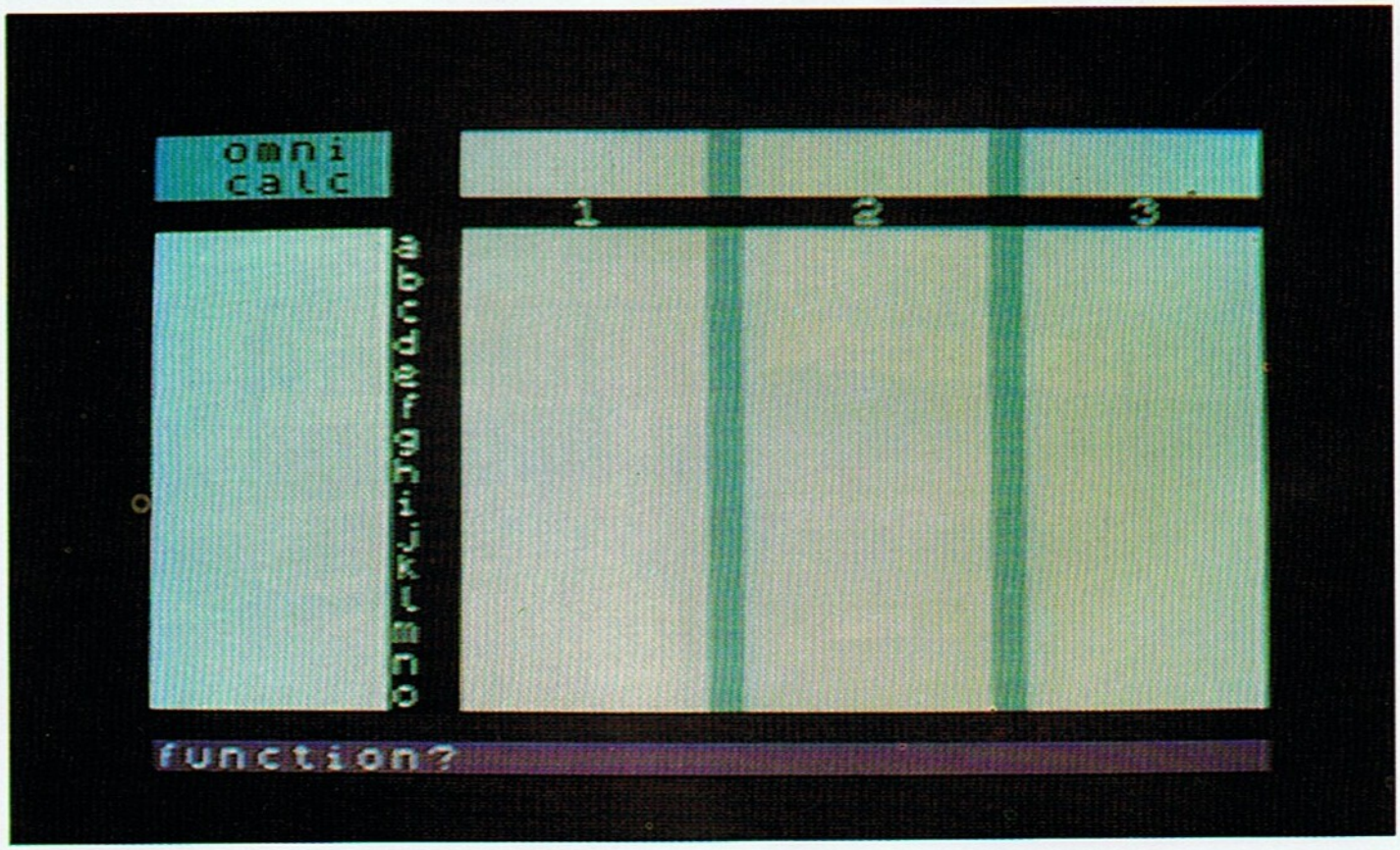


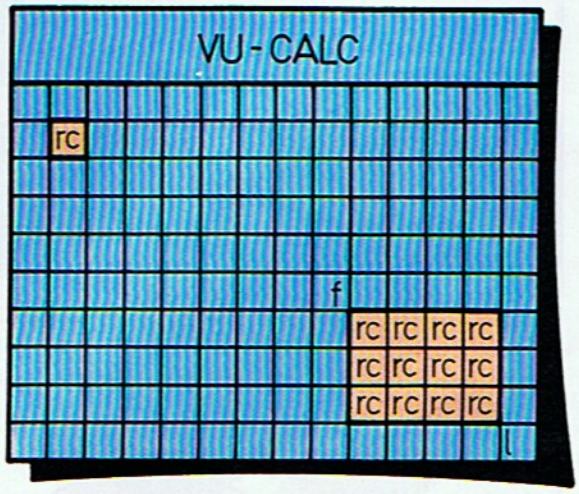
La opción F permite además examinar los registros del fichero, indicando qué registros deben visualizarse o imprimirse. Pulsando **N** es posible acceder al registro siguiente, pulsando **D** eliminar el registro en pantalla, pulsando **A** modificar su contenido, regresando al Menú pulsando **M**. La opción de búsqueda por campo permite localizar cualquier registro rápidamente sin necesidad de visualizar la totalidad del fichero. Debe señalarse qué campo se desea buscar. Las opciones disponibles a partir de entonces dependen de si se trata de un campo alfanumérico o numérico. En el primer caso puede escogerse entre la búsqueda de un campo determinado o parte de un campo, buscar un valor determinado, valores mayores que uno dado, menores, o entre límites. La hoja de cálculo permite incluir encabezamientos para las columnas (opción C), copiar fórmulas de una casilla a otra (opción D), mostrar la fórmula contenida en una celda (opción E), introducir una nueva fórmula (opción F), cargar el programa de gráficos (opción G), colocar el cursor al principio de la hoja (opción H), listar las fórmulas de la hoja (opción L), restaurar número anterior



Para mejorar la salida impresa de VU CALC, los números pueden aparecer justificados a izquierda o derecha con el comando #F.

La realización de totalizaciones es una de las características más empleadas de las hojas electrónicas.





Los parámetros del comando #R en VU CALC indican la zona de repetición de una fórmula.

(opción O), imprimir (opción P), inicializar (opción Q), introducir nombre de fila (opción R), grabar fichero (opción S), introducir título hoja (opción T), recalcular (opción U), borrar una fórmula (opción W) y borrar los valores de la hoja (opción Z).

El programa de gráficos requiere la organización previa de una archivo con la hoja de cálculo. Hecho esto, se puede obtener la interpretación de los datos contenidos en la hoja en forma analógica.

MINI OFFICE está preparado para su uso con la impresora ZX PRINTER, o bien con *interfaces* de impresora que no ocupen memoria RAM, como la impresora SEIKOSHA GP50.

El programa MINI OFFICE constituye una gestion integrada que incorpora procesador de textos, base de datos, hoja electrónica y generador de gráficos.

LOS INDICADORES

Hasta este capítulo los nuevos conceptos se acumularon uno tras otro, sin una explicación que clarificara su aparición. Como ya sabemos no era capricho: todo lo contrario.

Sin ellos sería imposible obtener provecho de todo lo siguiente. Es más, tengamos en cuenta que el significado de muchas instrucciones está relacionado con varios de estos conceptos, y sería complicado además de repetitivo establecer sin su conocimiento previo, un método que expusiera claramente qué es lo que hace determinada orden.

Por estas razones, nuestro esfuerzo inicial, como en adelante comprobaremos, quedará sobradamente recompensado y no nos cansaremos de recordar, aun a costa de resultar redundantes, la importancia fundamental de la lectura atenta y detallada de los capítulos anteriores antes de seguir adelante.

Como observaremos a partir de ahora, además de discutir el objetivo de una determinada instrucción y su funcionamiento, encontraremos otros datos de interés como son los referentes al tiempo de operación y las posibles inferencias de ésta con el registro F.

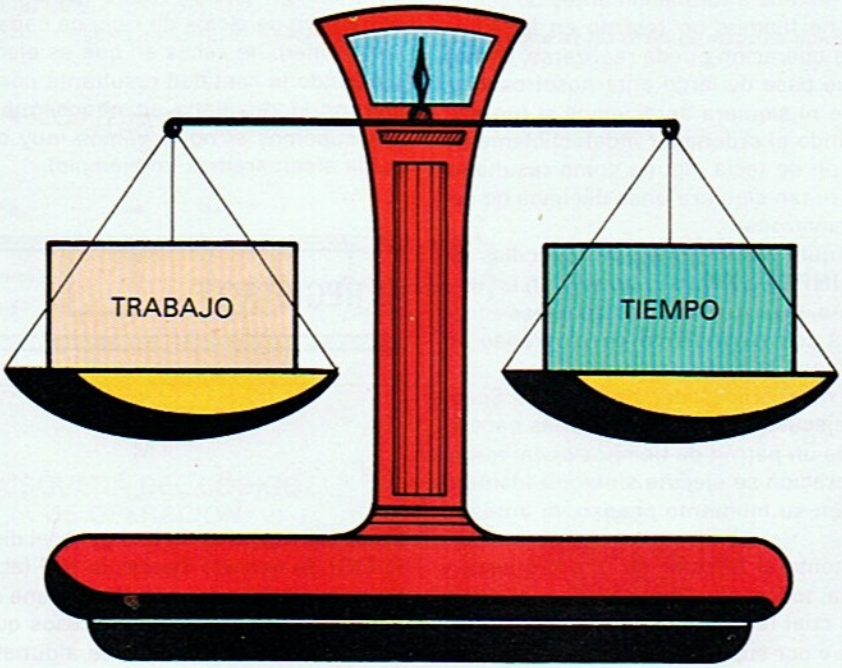
Aunque estos temas fueron tratados también anteriormente, no está de más profundizar sobre ellos con tal de familiarizarnos con su significado, pues su participación en cualquier rutina en C/M es de importancia capital.

CUESTION DE TIEMPO

El factor «tiempo», cuando se trata de programas escritos en lenguaje máquina, es fundamental. Por dos razones: la primera justifica si nuestra inversión de trabajo al construir una rutina en concreto ha sido acertada, es decir, si esta es lo suficientemente rápida como para emplearla en lugar de una idéntica codificada en BASIC.

Si el proceso seguido es correcto, la considera-

Antes de decidirse por la ejecución de una rutina en C/M es importante sopesar si el trabajo está en equilibrio con el beneficio obtenido.

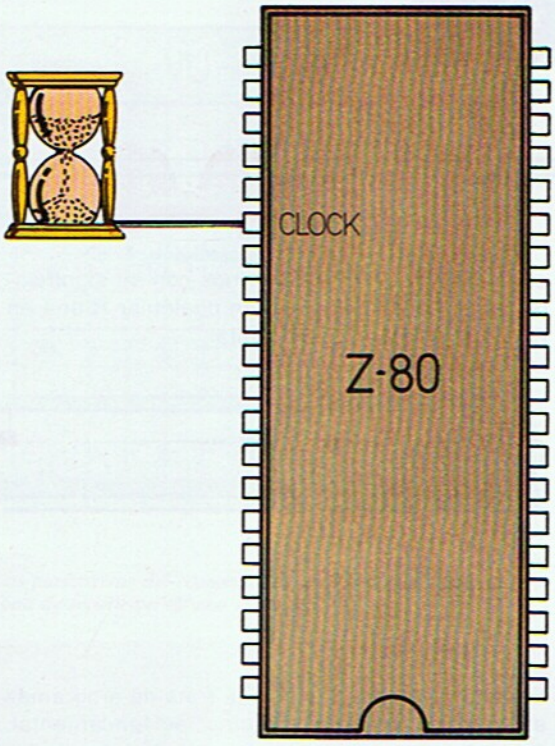


¡!

El factor tiempo, cuando se trata de programas escritos en C/M es fundamental.

*

Al utilizar el cargador hexadecimal emplea el modo mayúsculas para teclear los datos en la sentencia **DATA**.



A través del pin número 6 recibe el A 80 la señal de reloj.

¡!

El registro F tiene como misión revelarnos determinados acontecimientos que se producen tras la ejecución de una instrucción.

*

Si tras ejecutar una instrucción, el bit de CERO (Z) está a cero, quiere decir que el resultado de tal operación NO fue cero.

*

El Z80-A está capacitado para funcionar con una frecuencia de hasta 4 MHz.

ción anterior se cumplirá en la inmensa mayoría de los casos, pero a veces, no siempre constituirá una ventaja, sino tal vez, todo lo contrario.
Imaginemos un programa que necesite recibir del teclado determinada información antes de seguir adelante. Si no fijamos un retardo en la rutina para que esta operación pueda realizarse, lo más seguro es que pase de largo ante nosotros a tal velocidad que ni siquiera apreciemos si fue ejecutada, tomando el ordenador indefectiblemente la no pulsación de tecla alguna como resultado, al no darnos ni tan siquiera unas décimas de segundo para reaccionar.
Recordemos que instrucciones tan sencillas en BASIC como **INPUT** o **PAUSE**, no existen en lenguaje máquina, y para simularlas debemos construir la rutina correspondiente considerando en ella el tiempo que la necesitamos activa.
Para que el microprocesador de nuestro Spectrum pueda ejecutar todas sus funciones necesita disponer de un patrón de tiempos de tal forma que cada operación se ejecute sincronizadamente, es decir, en su momento preciso; ni antes ni después.
Es algo así como el director de una orquesta y bajo su batuta, todos los componentes realizan la labor para la cual fueron construidos sin dar lugar a errores y por supuesto, sin desafinar.

Este cronometraje lo efectúa un oscilador de cuarzo, el cual entrega para nuestro micro tres millones y medio de señales cada segundo (técnicamente se dice que está regulado a 3,5 MHz), denominándose cada una ciclo de reloj o estado, recibidas en el Z80-A a través de su pin número 6 *(CLOCK,* reloj).
En realidad el Z80-A está capacitado para funcionar hasta con una frecuencia de 4 MHz, pero los fabricantes eligieron la anteriormente señalada al considerar que si se llevaba al límite podría constituir una posible fuente de errores.
Una sencilla división nos muestra el tiempo empleado por un ciclo de reloj:

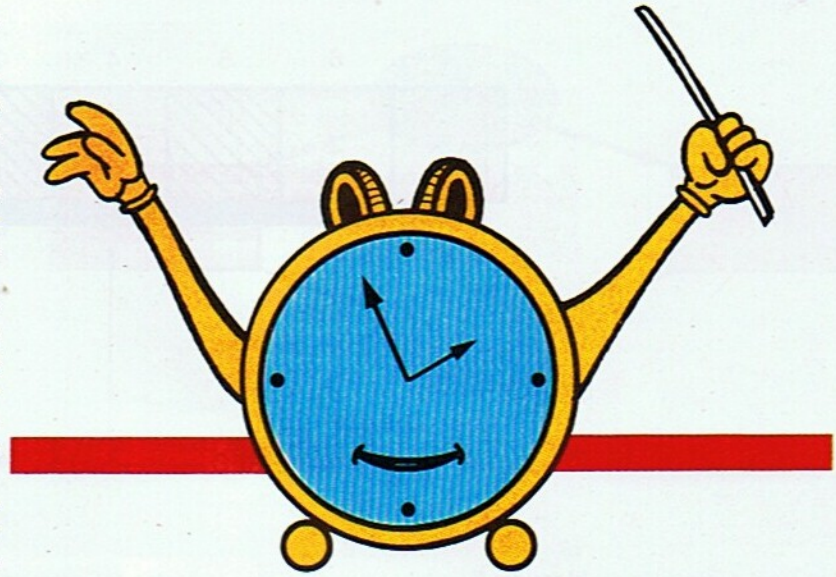
$$\frac{1}{3500000} = 2{,}85 \times 10^{-7}$$

es decir, 285 nanosegundos (segundos divididos por 10^9).
Se conoce como ciclo de instrucción al tiempo utilizado por el microprocesador en ejecutarla completamente. Dicha operación está compuesta por otras más elementales, denominándose al tiempo empleado en cada una de ellas ciclo de máquina.
Ha de quedar claro para todos que estos tiempos son diferentes según la instrucción particular o la operación elemental en cuestión. Así, por ejemplo, el ciclo de máquina durante el cual el microprocesador busca el código de operación de la instrucción toma 4 ciclos de reloj, mientras que las operaciones de lectura o escritura en la memoria (ciclos de memoria) ocupan 3 de reloj.
Así pues, para determinar el tiempo de ejecución de una determinada rutina bastará con conocer el número de ciclos de reloj de cada instrucción y el número de veces en que es ejecutada. Multiplicando la cantidad resultante por 285, obtendremos el resultado en nanosegundos (no nos preocupemos si no lo vemos muy claro. En seguida efectuaremos un ejemplo).

EL REGISTRO F

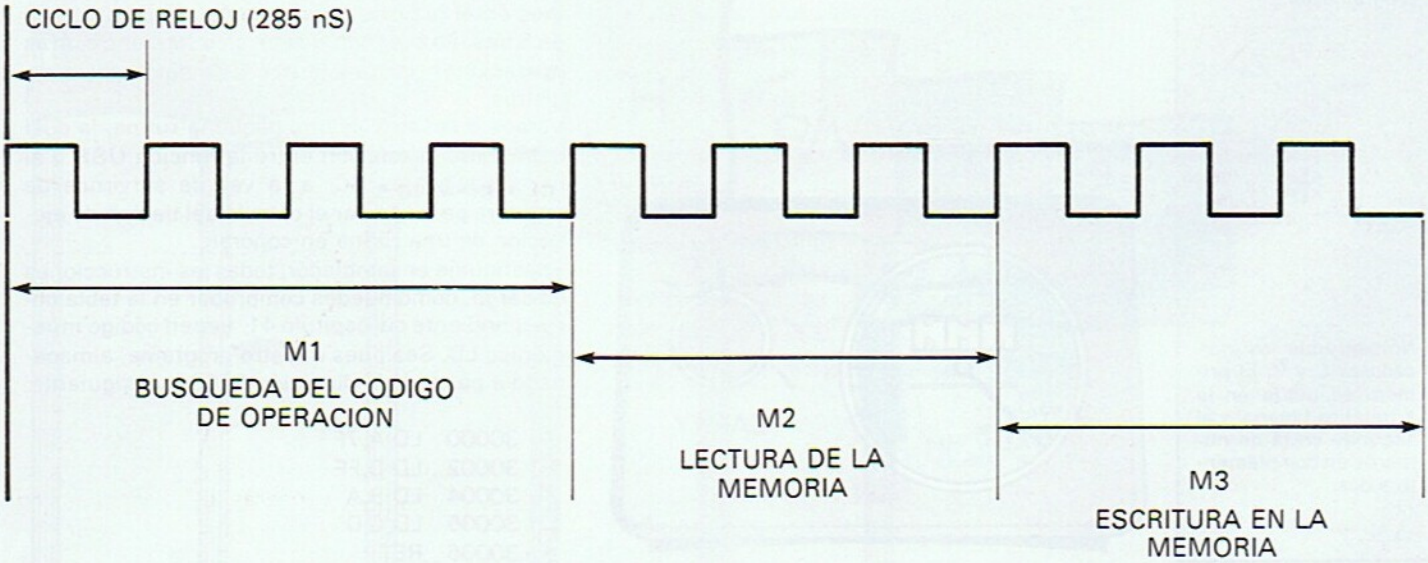
Entre los registros de uso general disponibles en la CPU de nuestro Spectrum el F (abreviatura de Flags, indicadores o banderas) tiene como misión revelarnos determinados sucesos que se producen durante la ejecución de algunas instruccio-

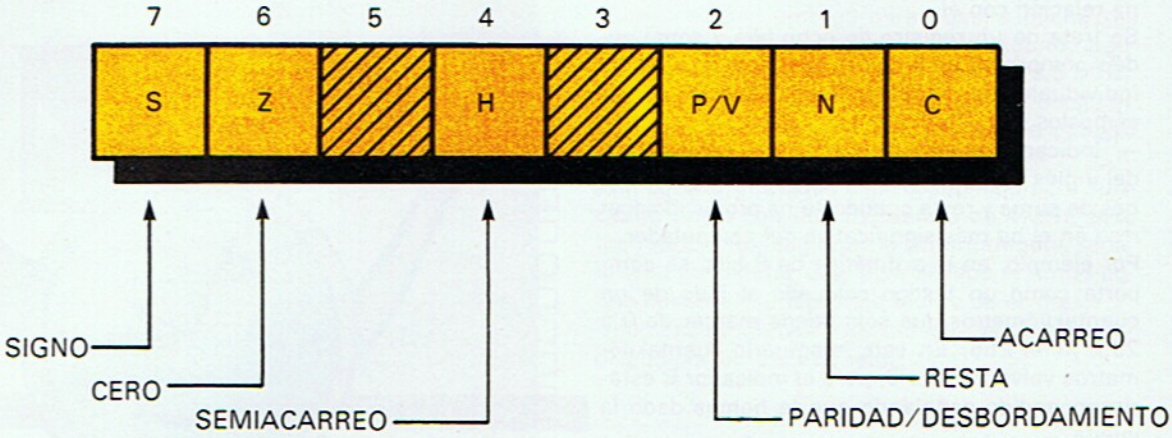
nes, las cuales aparentemente, no tienen ninguna relación con él.
Se trata de un registro de ocho bits y como podéis comprobar en la figura cada uno de sus bits individuales tiene un significado especial. Analicémoslos con detenimiento:
— Indicador de arrastre o acarreo C (abreviatura del inglés *Carry):* su contenido varía en operaciones de suma y resta cuando se ha producido acarreo en el bit más significativo del acumulador.
Por ejemplo, en la aritmética de 8 bits, se comporta como un testigo colocado al lado de un cuentakilómetros que sólo puede marcar de 0 a 255. ¿Y el 256? En este imaginario cuentakilómetros volvería a ser 0, pero el indicador C estaría encendido señalando que le hemos dado la vuelta.
— Indicador de cero Z (del inglés *Zero):* son muchas las instrucciones que afectan a su valor, por ejemplo, las que modifiquen el contenido del acumulador, las operaciones sobre bits individuales, las comparaciones, etc.
Su forma de trabajar resulta un tanto confusa si consideramos que cuando encontramos en su lugar correspondiente del registro F un 0, quiere decir que el resultado de la operación efectuada NO FUE CERO.
— Indicador de signo S *(Sign,* signo en inglés): el propósito de este bit es señalar si el resultado de una operación que involucre números en complemento a dos es positiva (bandera a 0) o negativa (bandera a 1).
Por tanto, no es otra cosa que una copia del bit más significativo de un número expresado en complemento a dos.
— Indicador de paridad/desbordamiento P/V *(Parity/oVerflow):* a este bit le están encomendadas dos misiones diferentes. La primera se encarga de señalar si, por ejemplo, tras una instrucción de rotación, desplazamiento o lógica, el número de bits que están en 1 lógico es par (indicador a 1) o impar (indicador a 0).
La segunda consiste en revelar si se ha producido desbordamiento en operaciones aritméticas de números expresados en complemento a dos. Tal es el caso que se presenta cuando la suma de



El reloj es el director de la orquesta informática interior al Spectrum.

Cronograma de los ciclos de máquina típicos en el Z80-A.





El registro F.

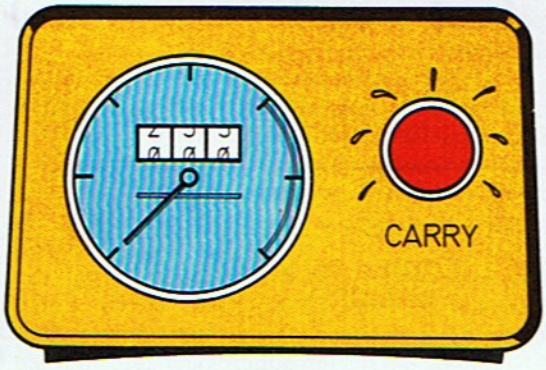
dos números positivos es negativa o la de dos negativos resulta positiva.
En estas circunstancias, el indicador V se coloca a 1, y en caso de no existir sobrepasamiento el valor encontrado en la posición correspondiente del registro F será 0.
— Indicador de semiacarreo H *(Half carry)*: se utiliza solamente en la aritmética codificada en decimal (BCD), la cual será comentada en su momento. Su funcionamiento es similar a como lo hacía el de arrastre, pero con la salvedad de que nuestro cuentakilómetros en este caso tendrá solamente 4 bits a diferencia con el anterior, el cual disponía de 8.
— Indicador de resta N: al igual que el anterior, solamente se emplea en la aritmética BCD, colocándose a 1 cuando la última operación efectuada fue una resta.
Como observaremos, sólo 6 de los 8 bits que componen el registro F tienen significado para nosotros. Los otros dos no tienen ningún interés para el programador.
No quiere esto decir que su valor no sea modificado durante la ejecución de una rutina determinada. Al contrario, la CPU los utiliza como apoyo para completar determinadas labores internas.
Antes de adentrarnos definitivamente en los diferentes grupos de instrucciones, hemos de recalcar la importancia que como nexo de unión entre BASIC y C/M tienen el par de registros BC, dentro del sistema operativo de nuestro Spectrum.
Es precisamente su contenido en el momento de retornar desde una rutina en código máquina, lo que obtenemos tras ejecutar la función **USR**. Comprobémoslo con algunos ejemplos.
El primer grupo de instrucciones que analizaremos en el próximo capítulo es el grupo de carga de 8 bits. Es el encargado de ejecutar, entre otras operaciones, la transferencia de datos entre registros.
Vamos a desarrollar una pequeña rutina, la cual demuestre la relación entre la función **USR** y el par de registros BC, a la vez de servirnos de muestra para ilustrar el cálculo del tiempo de ejecución de una rutina en concreto.
En lenguaje ensamblador, todas las instrucciones de carga, como puedes comprobar en la tabla correspondiente del capítulo 41, tienen código mnemónico LD. Sea pues nuestro programa, almacenado a partir de la dirección 30000, el siguiente:

```
30000  LD A,7F
30002  LD D,FF
30004  LD B,A
30005  LD C,D
30006  RET
```

Sólo 6 de los 8 bits del registro F tienen utilidad práctica desde el punto de vista de la programación.

No confundir los indicadores C y V. El primero se utiliza en la aritmética binaria, y el segundo en la de números en complemento a dos.

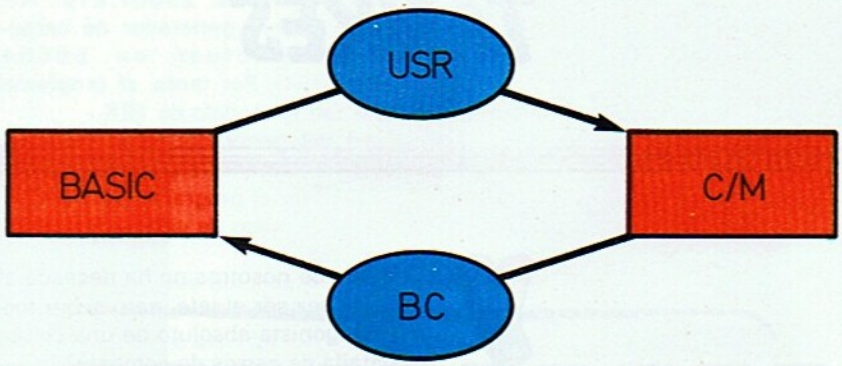
Indicador de arrastre o acarreo C.



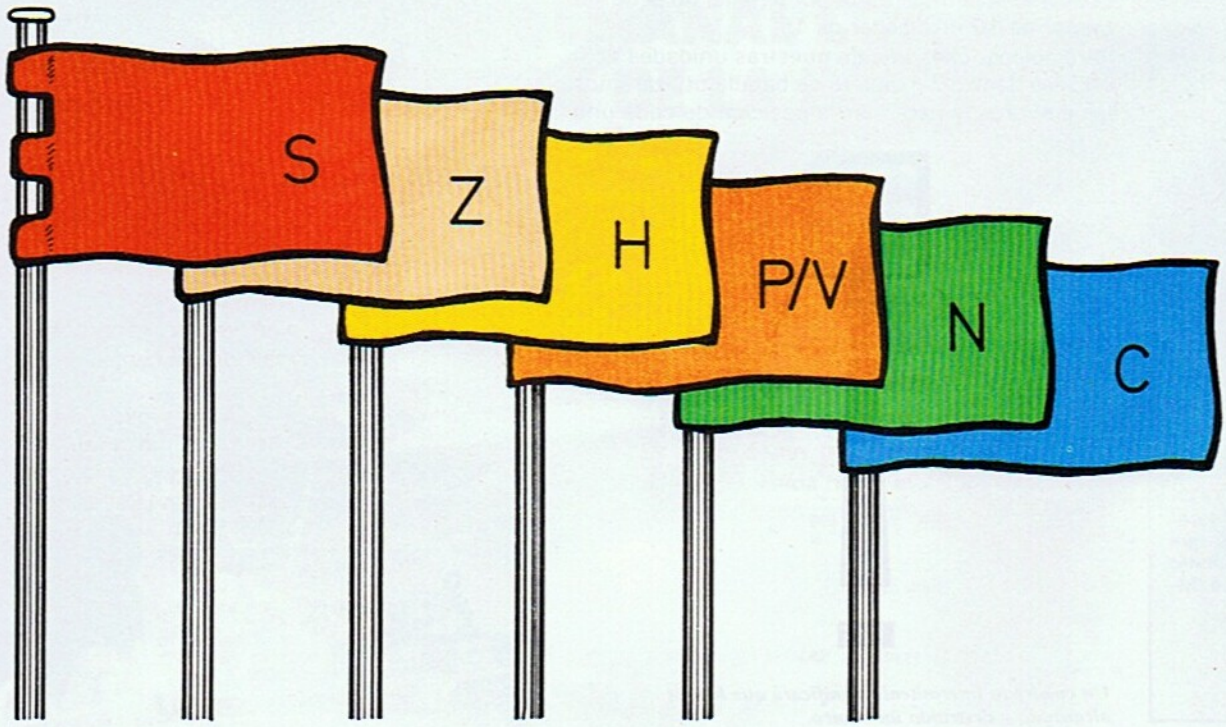
¿Qué es lo que hace? Las dos primeras instrucciones almacenan en los registros A y D los valores hexadecimales 7F y FF respectivamente. A continuación, estos valores son transferidos a los registros B y C. Finalmente, mediante la instrucción RET se devuelve el control desde el C/M al BASIC.
Para almacenarla en la memoria del ordenador puedes utilizar el cargador de código máquina aparecido anteriormente en esta misma sección. Carga éste y añade la siguiente línea:

10 DATA "3E 7F 16 FF 47 4A C9 **" ,812

Ejecuta el programa mediante **RUN** contestando a la pregunta sobre dirección de ubicación con 30000. Si el cargador indica algún error, repasa los datos anteriores, hasta que todo vaya bien. Teclea entonces, como comando directo **CLS: PRINT USR 30000.**
Tras pulsar **ENTER** aparecerá sobre la pantalla 32767, el cual es efectivamente, la versión decimal del número 7FFF hexadecimal, almacenado en el par BC.
Para terminar, calculemos el tiempo empleado en ejecutarse esta pequeña rutina. En las dos primeras instrucciones, el microprocesador debe acceder dos veces a la memoria del ordenador, para leer en primer lugar el código de operación (4 ciclos de reloj), y posteriormente, el dato a ser introducido en el registro correspondiente (3 ciclos).
En las dos siguientes le basta con el código de operación para ejecutarlas, es decir, cada una de ellas empleará 4 estados. La instrucción RET, la cual será discutida más extensamente cuando tratemos las instrucciones de retorno, toma 10 ciclos en su ejecución.
En definitiva 2×7+2×4+10=32 ciclos de reloj, o lo que es lo mismo, 9120 nanosegundos. ¡Menos de 10 millonésimas de segundo en efectuar todo lo anterior!



El par de registros BC constituye el nexo de unión entre BASIC y C/M.



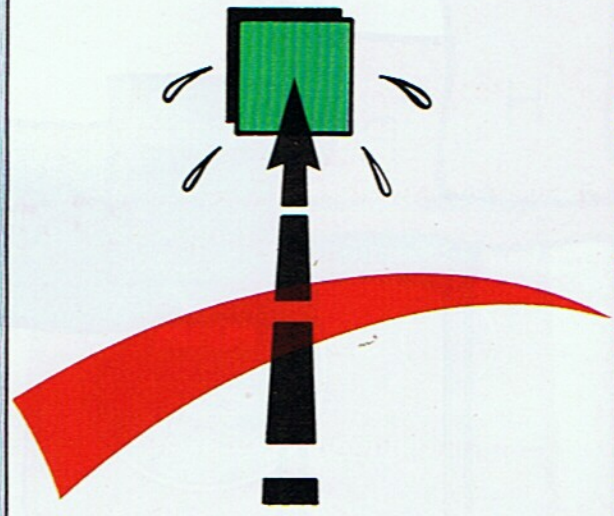
El registro F está formado por 6 banderas de interés para nosotros.

TANKES

Y quien de nosotros no ha deseado alguna vez ser el jefe, pero sobre todo protagonista absoluto de una colosal batalla de carros de combate?
Con este programa podremos, sin lugar a dudas, emprender una superaventura bélica, teniendo por enemigo la escuadra blindada del Spectrum.

EL PROGRAMA

Una vez el listado del programa ha sido correctamente introducido en la memoria de nuestro ordenador y ejecutado, surgirán en nuestra pantalla un par de mensajes de captación de datos. El primero de ellos nos pedirá que introduzcamos nuestras iniciales; con ésto, nuestro Spectrum se hace una ligera idea de con quien tiene que enfrentarse. El segundo mensaje de captación nos requiere el número de carros de combate que queremos entren en batalla. El número de tanques que podemos reunir por batalla, no debe ser menor de 10 ni exceder de 16.
Para colocar cada una de nuestras unidades acorazadas dentro del tablero de batalla utilizaremos los cursores, y para fijar la posición de cada uno pulsaremos la tecla donde está representado el número cero.
Una vez hemos posicionado todos los tanques, nuestro serio oponente Spectrum hará lo propio con los suyos y la batalla comenzará.
A la hora de efectuar un disparo, debemos seguir las mismas reglas de pulsación de teclas que cuando situamos nuestros carros de combate. Con las teclas 6 y 7 elegiremos nuestro campo de tiro. Con el 5 y 8 daremos más alcance a nuestro disparo o corregiremos, y con el 0 efectuamos el cañonazo.
Un cuadrado intermitente significará que hemos alcanzado un blanco y, como consecuencia, des-



Un cuadrado intermitente significará que hemos alcanzado y destruido un blanco.

¡!

Para la grabación utilizaremos el comando **SAVE "TANQUES" LINE 10.** La autoejecución en la línea 10 propicia la carga automática del generador de caracteres.

*

El juego de caracteres utilizado para este programa aparece bajo el nombre de ROTULACION en nuestro anterior programa IMPRENTA.

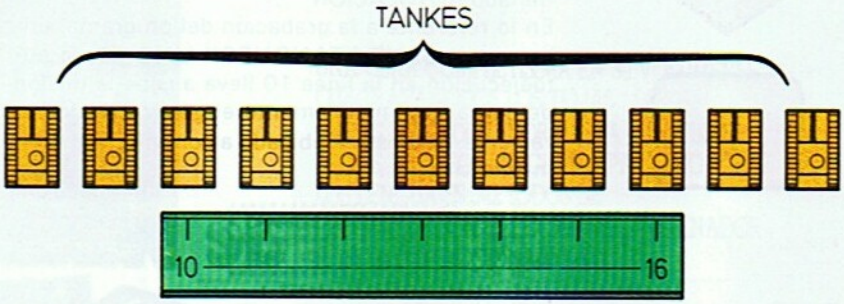
*

El juego de caracteres ROTULACION deberá ubicarse a continuación del programa principal en la misma cinta. Para su grabación deberemos seguir las instrucciones que aparecen en nuestro anterior programa IMPRENTA.

truido. Si nuestro disparo no hubiese encontrado el objetivo deseado, nuestro ordenador lo marcaría en la pantalla de una forma diferente: delimitación de un cuadrado.

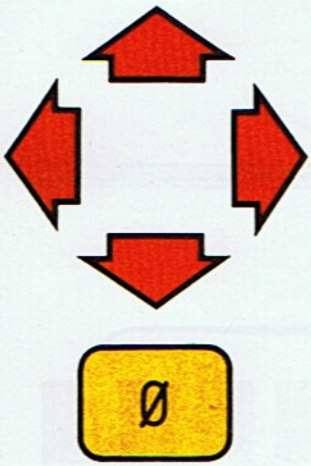
La batalla finaliza cuando la totalidad de los tanques de alguno de los dos bandos ha sido destruido. En caso de que seamos los perdedores, nuestro enemigo, no por ello menos galante, nos mostrará aquellos blancos que no pudimos destruir.

El programa utiliza un nuevo generador de caracteres para dar un poco más de ambientación al juego. Los **POKEs** que efectúa el Spectrum para acceder a este generador, son los siguientes:



El número de tanques que podemos reunir por batalla, no debe ser menor que diez ni exceder de dieciséis.

ALFONSO MENDEZ



POKE 23606,102 y **POKE 23607,216**. Así pues, la carga del nuevo generador de caracteres se debe efectuar en 55654 (102+256*217=55654). Por tanto, el programa no es compatible con el modelo de 16K.

Para la adopción del generador de caracteres, procederemos por el sistema explicado en esta misma sección al tratar el programa IMPRENTA. Hemos de emplear el juego de caracteres denominado ROTULACION.

En lo referente a la grabación del programa, utilizaremos **SAVE "TANQUES" LINE 10**; la autoejecución en la línea 10 lleva a cabo la misión de cargar automáticamente el generador de caracteres, que estará ubicado a continuación en la misma cinta.

```
 10 REM ************************
 20 REM * J.M.MAYORAL SERRANO *
 30 REM ************************
 40 REM * TANKES     (C) 1985 *
 50 REM ************************
 60 INK 9: BORDER 1; PAPER 1
 70 CLS
 80 PRINT AT 10,4;' UN MOMENTO, POR FAVOR '
 90 PRINT AT 12,6; FLASH 1;;' GRAFICOS USUARIO '
100 GO SUB 1320: CLS
110 LET SWC=0
120 LET CP=1: LET V1=0: LET V2=0
130 DIM G$(2,5): LET G$(2)=' YO! '
140 LET JK=0: LET G=0
150 PRINT AT 11,3;' INTRODUCE TUS INICIALES '
160 INPUT  LINE G$(1)
170 PRINT AT 11,3;'                          '
180 PRINT AT 14,0;'CUANTOS TANKES QUIERES ? (10-16)'
190 INPUT N
200 IF N<10 OR N>16 THEN  GO TO 190
210 LET BDM=0
220 CLS
230 GO SUB 1230
240 DIM P(2)
250 GO SUB 1090
260 DIM A(64)
270 DIM B(64)
280 FOR K=2 TO 16 STEP 2
290 FOR W=4 TO 18 STEP 2
300 PRINT AT K,W; INK 5;'a'
310 NEXT W
320 NEXT K
330 GO SUB 1210
340 PRINT AT 13,23;'PULSE   ';AT 14,23;'5-6-7-8 ';AT
15,23;'TANKE:
350 FOR K=1 TO N
360 LET POS=2
370 PRINT  INK 6;AT POS,2;'A'; INK 7;AT 2,20;'C'
380 PRINT AT 16,23; BRIGHT 1; FLASH 1;K
390 LET M$=INKEY$
400 IF M$<>'5' AND M$<>'6' AND M$<>'7' AND M$<>'8' T
HEN  GO TO 390
410 LET M=VAL M$: IF M=8 OR M=5 THEN  GO TO 480
420 LET PZ=POS
430 LET POS=POS+(2 AND M=6)-(2 AND M=7): IF POS<2 TH
EN  LET POS=4
440 IF POS>16 THEN  LET POS=14
450 PRINT  INK 6;AT POS,2;'A';AT PZ,2;' '
460 BEEP .06,15
470 GO TO 390
480 PRINT  INK 4; BRIGHT 1;AT POS,4;'B': LET PZ=4
490 LET M$=INKEY$: IF M$='0' THEN  GO TO 580+(220 AN
D BDM=1)
500 IF M$<>'5' AND M$<>'8' THEN  GO TO 490
510 LET PT=PZ
520 LET PZ=PZ+(2 AND M$='8')-(2 AND M$='5'): IF PZ<4
THEN  LET PZ=6
530 IF PZ>18 THEN  LET PZ=16
540 GO SUB 1100
550 PRINT  INK 4; BRIGHT 1;AT POS,PZ;'B'
560 LET PY=PY-(M$='8')+(M$='5'): GO SUB 1040
570 GO TO 490
580 GO SUB 1100
590 IF A(PY)=1 THEN  GO TO 490
600 LET SWC=1
610 LET A(PY)=1
620 GO SUB 1040
630 BEEP .12,6: BEEP .12,0
640 PRINT AT POS,2;' '
650 NEXT K
660 LET POS=2
670 PRINT AT 19,1; FLASH 1; BRIGHT 1;'UN MOMENTO, PO
R FAVOR'
680 FOR K=1 TO N
690 LET NC=INT (RND*64)+1: IF A(NC)=1 OR A(NC)=2 THE
N  GO TO 690
700 LET A(NC)=2: LET B(NC)=1: NEXT K
710 PRINT  PAPER 2;AT 19,1;'                          ';A
T 19,1;'TANKES: ';N;'  BATALLA ';CP
720 FOR K=13 TO 19: PRINT AT K,23;'          ': NEXT K
: PRINT AT 14,23;'TURNO DE'
730 PRINT  INK 6;AT POS,2;'A'
740 LET G=G+1: IF G=3 THEN  LET G=1
750 PRINT AT 15,23; BRIGHT 1;G$(G)
760 PRINT  INK 6;AT POS,2;' ';AT 2,2;'A'
770 IF G=2 THEN  GO TO 900
780 LET BDM=1
790 GO TO 390
800 LET SWC=1
810 GO SUB 1100
820 IF A(PY)=0 THEN  LET A(PY)=5: BEEP .3,8: BEEP .5
,-30: GO SUB 1040: GO TO 740
830 IF A(PY)<>2 THEN  GO TO 740
840 LET P(1)=P(1)+1
850 GO SUB 1090
860 LET JK=1: LET A(PY)=4: GO SUB 1040
870 FOR Z=1 TO 3: BEEP .3,7: BEEP .13,4: NEXT Z: BEE
P .5,0
880 LET SWC=1: GO SUB 1040: IF P(1)=N THEN  GO TO 11
10
890 GO TO 740
900 PAUSE 50
910 LET T=INT (RND*64)+1: IF B(T)=1 THEN  GO TO 910
920 LET SP=INT (T/8): IF SP*8<T THEN  LET SP=SP+1
930 LET POS=SP*2: FOR X=2 TO POS STEP 2: PRINT AT X,
20;'C';AT X-1-(X<>2),20;' ': BEEP .2,15: BEEP .3,0: N
EXT X
940 LET ED=0: LET CONT=SP*8-T+1
950 FOR X=18 TO 4 STEP -2: PRINT AT POS,X; INK 4; BR
IGHT 1;'B': BEEP .2,30
960 GO SUB 1220: LET PT=X: GO SUB 1040: LET CONT=CON
T-1: IF CONT<=0 THEN  GO TO 980
970 LET ED=ED+1: NEXT X
980 LET B(T)=1: IF A(T)=1 THEN  GO TO 1010
990 BEEP .2,0: BEEP .3,3
1000 PRINT AT POS,20;' ';AT 2,20;'C': LET POS=2: GO T
O 740
1010 LET JK=1: LET P(2)=P(2)+1: GO SUB 1090: GO SUB 1
040: FOR W=1 TO 3: BEEP .2,-3: BEEP .4,3: NEXT W
1020 BEEP .4,-35: LET A(T)=3: GO SUB 1040: IF P(2)=N
THEN  GO TO 1170
1030 GO TO 1000
1040 BRIGHT (JK=1): FLASH (JK=1)
1050 PRINT AT POS,(PT AND SWC=0)+(PZ AND SWC=1); INK
5;('B' AND A(PY)=0); INK 6;('D' AND A(PY)=1); INK 5;(
'B' AND A(PY)=2); INK 3; BRIGHT 1;('X' AND A(PY)=3);
INK 2; BRIGHT 1; FLASH 1;('E' AND A(PY)=4); FLASH 0;
INK 6;('E' AND A(PY)=5)
1060 BEEP .02,-3+(6 AND BDM=1)
1070 FLASH 0: BRIGHT 0
1080 LET JK=0: LET SWC=0: RETURN
1090 PRINT AT 6,29;V1;AT 7,29;V2;AT 10,29;P(1);AT 11,
29;P(2): RETURN
1100 LET PY=((POS/2-1)*7+POS/2)+((PZ/2)-2): RETURN
1110 GO SUB 1090: GO SUB 1210: PRINT AT 14,23; INK 2;
 FLASH 1;' TU ';AT 15,23;' GANAS ': PAUSE 100
1120 LET V1=V1+1: LET CP=CP+1
1130 PRINT  INK 6;AT 21,2;'DESEAS SEGUIR JUGANDO ?(S/
N)'; INK 0
1140 IF INKEY$='S' THEN  GO TO 210
1150 IF INKEY$='N' THEN  GO TO 10000
1160 GO TO 1140
1170 GO SUB 1090: GO SUB 1210: PRINT AT 14,23; FLASH
1;'GANO YO'
1180 LET X=0: FOR K=2 TO 16 STEP 2: FOR W=4 TO 18 STE
P 2: LET X=X+1: IF A(X)=2 THEN  PRINT AT K,W; FLASH 1
; BRIGHT 1;'D': BEEP .3,5: BEEP .3,-40
1190 NEXT W: NEXT K
1200 LET V2=V2+1: LET CP=CP+1: GO TO 1130
1210 FOR K=13 TO 19: PRINT AT K,23; BRIGHT 1;'
': NEXT K: RETURN
1220 LET PY=POS*4-ED: RETURN
1230 CLS
1240 FOR K=0 TO 31: PRINT AT 0,K;'a';AT 18,K;'a';AT 2
0,K;'a': NEXT K
1250 FOR K=0 TO 20: PRINT AT K,0;'a';AT K,22;'a';AT K
,31;'a': NEXT K
1260 PRINT AT 18,23;'          ';AT 8,23;'33333333';AT
12,23;'33333333';AT 4,23;'88888888'
1270 PRINT AT 0,5; INVERSE 1;' EL ALAMEIN ': INVERSE
0;AT 5,23;'TRIUNFOS';AT 6,23;G$(1);AT 7,23;'COMP.'
1280 PRINT AT 9,23;'IMPACTOS';AT 10,23;G$(1);AT 11,23
;'SPECT.';AT 2,24; BRIGHT 1;'TANKES'
1290 FOR K=27 TO 155 STEP 16: PLOT K,36: DRAW 0,128,0
: NEXT K
1300 FOR K=36 TO 164 STEP 16: PLOT 27,K: DRAW 128,0,0
: NEXT K
1310 RETURN
1320 DATA 'A',255,195,40,47,40,195,255,0,'B',0,0,60,6
0,60,60,0,0,'C',0,255,195,20,244,20,195,255,'D',0,128
,76,60,44,255,171,84
1330 DATA 'E',255,129,189,189,189,189,129,255,'F',255
,129,129,129,129,129,129,255
1340 FOR K=1 TO 6: READ C$: FOR W=0 TO 7: READ C: POK
E USR C$+W,C: NEXT W: NEXT K: RETURN
1350 CLEAR 55650: LOAD ''CODE
1360 POKE 23606,102
1370 POKE 23607,216
1380 RUN
```

LOGO

El LOGO es un lenguaje, que en sus algo más de diez años de existencia está teniendo importancia para la introducción en el mundo informático de los no iniciados. Precisamente Seymour Papert pensó en éstos al crear este lenguaje, concretamente en los niños. Una de las principales características que presenta es que aquel que no haya visto nunca un ordenador puede realizar programas en LOGO en muy poco tiempo, algo impensable en un lenguaje tan popular como el BASIC.

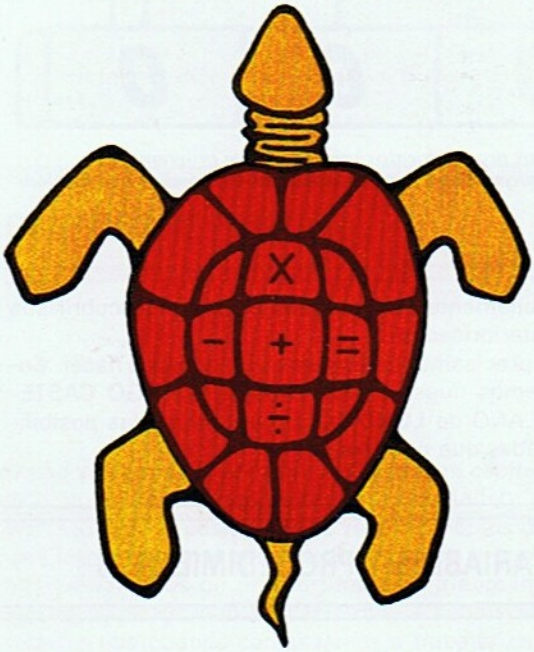
Por lo que llevamos dicho hasta ahora podríamos estar tentados a pensar que entonces este lenguaje es de segunda categoría y que sólo es «juego de niños». Al contrario, el LOGO es un lenguaje que presenta algunas características novedosas sobresalientes (como son los gráficos de tortuga que más adelante veremos) que han sido incorporadas actualmente en otros lenguajes considerados como más potentes.

Pero ¿cuáles son esas características que hacen que el LOGO sea un lenguaje de primera categoría pero apropiado para el no iniciado en este mundo? Se pueden resumir en las siguientes:

— Es un lenguaje informático que suele ser presentado en el idioma en que se expresa el usuario, por lo tanto éste tendrá menos problemas en aprenderse los comandos (mejor llamados primitivas) y en saber sus funciones.

— Lleva incorporado uno de los sistemas más simples pero al mismo tiempo más potentes para la realización de dibujos, los muy conocidos «gráficos de tortuga». Con ésto se pueden realizar dibujos muy complicados utilizando muy pocos comandos.

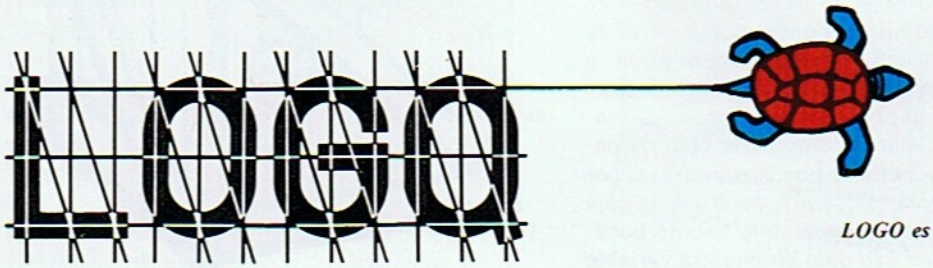
— Maneja de forma muy sencilla palabras y listas, que pueden estar formadas por números, palabras u otras listas, lo que puede ser equivalente en BASIC a las cadenas de caracteres y cuya potencia en LOGO hace que se vean diferentes.



Además de los manejos de listas y de los gráficos de tortuga, algunas versiones LOGO pueden realizar operaciones aritméticas y trigonométricas.

— Pertenece al grupo de los lenguajes llamados estructurados, es decir que utiliza procedimientos (bloques de programa que realiza una función concreta, equivalente en BASIC a las subrutinas llamadas por **GOSUB**) e incluso con posibilidad de recursividad, es decir, que un procedimiento se puede llamar a sí mismo.

— Si a todo esto se le añade que muchas versiones LOGO incorporan funciones normales en otros lenguajes como son las operaciones aritméticas y las razones trigonométricas, hace que nos encontremos con un lenguaje para sentarse tran-



LOGO es un lenguaje pensado para la educación.

¡!

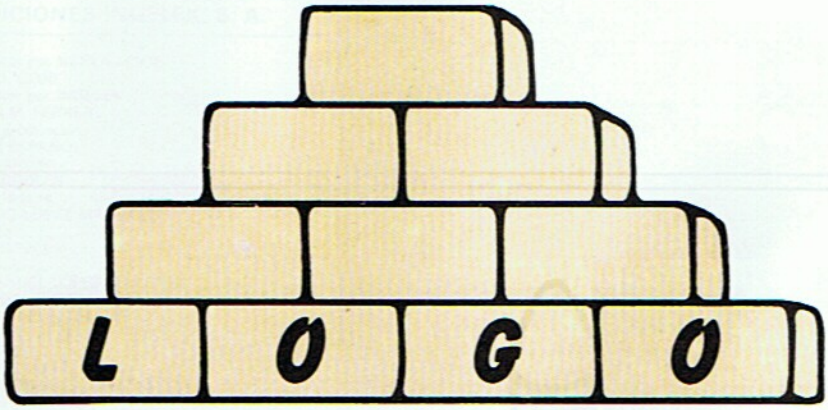
No hay que avisar previamente si vamos a trabajar con variables numéricas o alfanuméricas.

*

Para ejecutar un procedimiento se escribe su nombre como si se trátase de una primitiva incorporada.

*

Para trabajar con los gráficos se supone que tenemos una tortuga (ese triángulito que aparece en la pantalla) dotado de un lápiz, la cual dejará el trazo donde y cuando nosotros queramos.



Los procedimientos son bloques de programa independientes que cada uno de ellos realiza una función.

quilamente ante nuestro «micro» y descubrir sus interioridades.
Y precisamente esto es lo que vamos a hacer. Cogemos nuestro SPECTRUM y el LOGO CASTELLANO de **LCSI/SOLI** y bucear en las posibilidades que nos ofrece.

VARIABLES Y PROCEDIMIENTOS

¡!

El LOGO nos ofrece también la posibilidad de asignar a una variable palabras y listas.

*

Lleva incorporado uno de los sistemas más simples pero al mismo tiempo más potentes para la realización de dibujos: gráficos de tortuga.

En líneas generales el LOGO no trabaja de una manera diferente al BASIC con las variables, aunque sí cambia, como es de suponer, los comandos y la síntaxis. Podemos realizar el símil de suponer que una variable es como una caja en la cual guardamos la información que nos interesa. Para encontrar pronto lo que tenemos guardado en la caja, a ésta le ponemos un nombre que nos identifique el contenido de su interior, pues igualmente a una variable hay que ponerle un nombre para identificar lo que queremos guardar.
¿Ponemos un ejemplo? Supongamos que en nuestra caja queremos guardar los meses del año apuntados en una hoja de papel, por supuesto que el mejor título que podemos poner es el de «meses», y, entonces, siguiendo con el símil, a nuestra variable la llamaremos también «meses». Vamos a guardar el primer dato en nuestra «caja/variable». Para ello cogemos una hoja de papel y escribimos en ella lo que nos interesa, por ejemplo «enero», y la colocamos dentro de la caja con la precaución de dejar el dato escrito bocaarriba. Para guardar ese dato en nuestra variable «meses» en nuestro LOGO lo haremos de la siguiente manera:

? ASIGNA ''meses ''enero

Ahora ya tenemos la seguridad de que nuestro dato no se nos perderá y sabemos que siempre que acudamos a **''meses** nos encontraremos lo mismo que hemos guardado. En el caso de la caja nos basta con asomarnos y echar una mirada en su interior para saber qué es lo que tenemos guardado, pero, claro está, con el LOGO eso no lo podemos hacer. Le tendremos que decir a nuestro «micro» que por pantalla nos muestre el valor de la variable **''meses**:

? IMPRIME :meses

Y en pantalla aparecerá:

enero

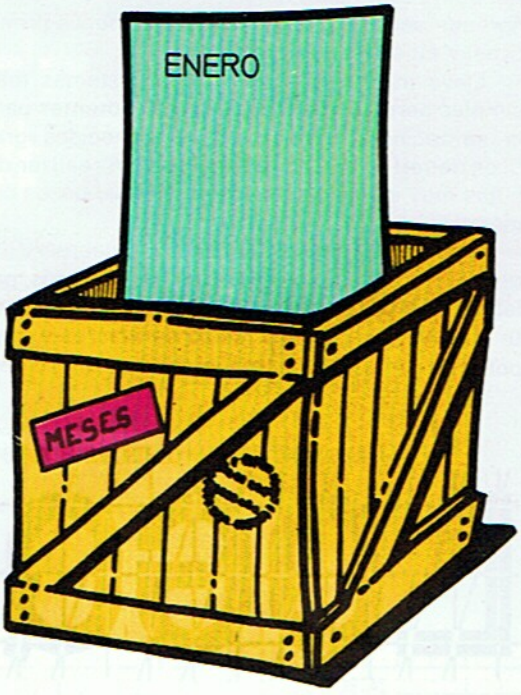
Sin embargo, el LOGO nos ofrece también la posibilidad de asignar a una variable no una sino una serie de palabras (lo que se conoce como listas). Sería el equivalente a que en nuestra caja introduzcamos una hoja en la que se han escrito varios meses. Veamos cómo se hace en LOGO:
?ASIGNA ''meses (enero febrero marzo abril)
Si entonces hacemos...

?IMPRIME :meses

En pantalla aparecerá, por tanto:

enero febrero marzo

A las variables se las puede considerar como cajas con su contenido especificado en el exterior, y en el cual introducimos la palabra a guardar.



Aunque aquí hemos trabajado con variables alfanuméricas, con números se hace exactamente igual, al LOGO no hay que avisarle previamente si vamos a trabajar con uno u otro tipo de variable. Por tanto, el lector comprenderá perfectamente las siguientes órdenes y sabrá qué aparecerá en la pantalla:

?ASIGNA ''dias 30
?ASIGNA ''dias (28 29 30 31)

NUESTRO PRIMER PROGRAMA

Sin embargo, programar así no aporta ninguna ventaja a no ser que podamos agrupar una serie de comandos para resolver un problema concreto. Es aquí donde entra en juego los procedimientos:

?PARA media
>IMPRIME (8+6)/2
>FIN

Aquí tenemos nuestro primer programa en LOGO que consta de un sólo procedimiento, al cual le hemos llamado «media». Para que nuestro intérprete LOGO distinga si estamos introduciendo un comando directo o definiendo un procedimiento se emplea la primitiva «PARA». Para indicar el final de dicho procedimiento se escribe «FIN», a lo cual el ordenador nos contestará, para indicar que se ha dado por enterado, con el siguiente mensaje:

media DEFINIDO

Para ejecutar nuestro procedimiento se escribe su nombre como si se tratase de una primitiva más incorporada:

?media
7

Sin embargo, este procedimiento nos sirve de muy poco puesto que sólo nos calcula la media de 8 y 6; para que sea verdaderamente útil y se pueda calcular la media aritmética de dos números cualesquiera deberíamos tener el siguiente procedimiento:

?PARA mediabis :prinum :segnum
>IMPRIME (:prinum + :segnum)+2
>FIN

Para ejecutar este procedimiento introduciremos el nombre y los valores que queramos tomen las variables en cada caso:

?mediabis 8 2
5

Es evidente que el valor 8 se asigna a «prinum» y el de 2 a «segnum».

Aunque el LOGO es un lenguaje para los que se inician en informática no debe ser considerado de segunda categoría.

GRAFICOS DE TORTUGA

Como ya dijimos al principio del presente capítulo, una de las características más sobresalientes del LOGO es la posibilidad de realizar dibujos de una forma bastante sencilla. Para ello se supone que disponemos de una tortuga (ese triangulito que aparece en medio de la pantalla señalando hacia arriba cuando comenzamos a trabajar con

En una variable también podemos guardar no sólo palabras sino una lista de ellas.

¡!

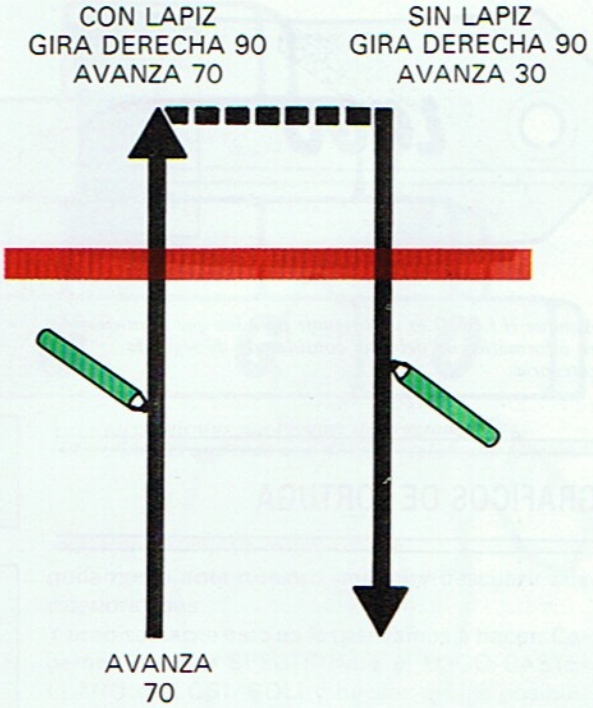
Podemos realizar una asignación a una variable de una lista formada a su vez por otra lista además de palabras.

*

Las listas podemos modificarlas (añadiendo palabras o listas), imprimirlas totalmente o imprimir cualquier parte de ella.

*

Maneja de forma muy sencilla palabras y listas, que a su vez pueden estar formadas, por números, palabras u otras listas.

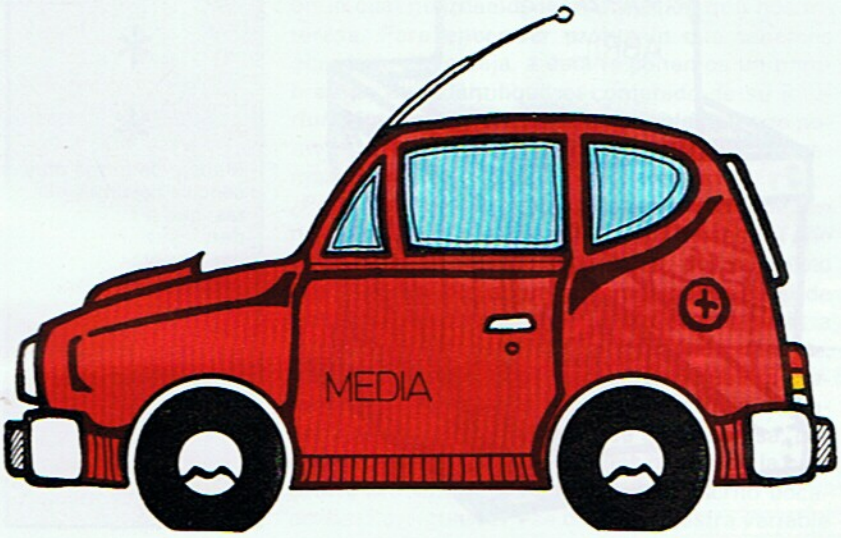


A nuestra tortuga le podemos decir que levante y baje el lápiz cuando nos convenga.

LOGO) muy bien amaestrada y dotada de un lápiz el cual dejará el trazo donde y cuando nosotros queramos. Así el siguiente programa le hacemos subir y bajar el lápiz a nuestra amiga:

?PARA paralelas
>AVANZA 70
>SINLAPIZ GIRADERECHA 90 AVANZA 30
>CONLAPIZ GIRADERECHA 90 AVANZA 70
>FIN

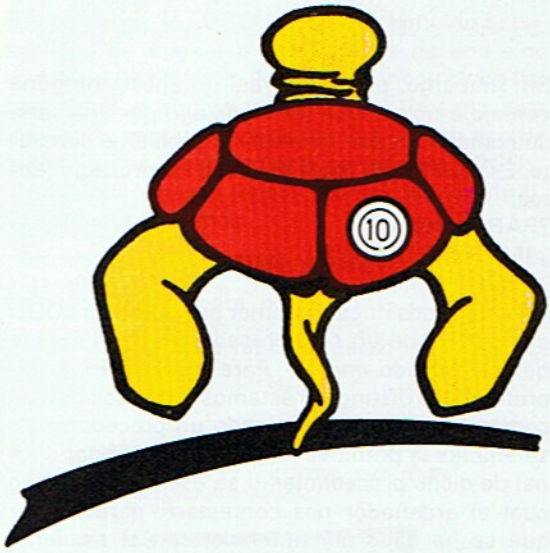
Para ejecutar un procedimiento sin variables basta con dar el nombre de dicho procedimiento.



En el manual del intérprete de LOGO viene un ejemplo de como hacer un cuadrado. Para evitar repetirnos vamos a ir más allá y aquí presentamos un procedimiento para realizar cualquier figura regular:

?PARA regular :lado
>BORRAPANTALLA
>ASIGNA "angulo 360/:lado
>REPITE :lado (AVANZA 10 GIRADERECHA :angulo)
>FIN

Como hemos podido comprobar, la comprensión de los procedimientos y de la función que ellos tienen es bien fácil ya que se utilizan primitivas escogidas de vocabulario usual, y ante las cuales nuestra amiga la tortuga se doblega y cumple la función encomendada con su lápiz a cuestas dejando la huella de su recorrido.



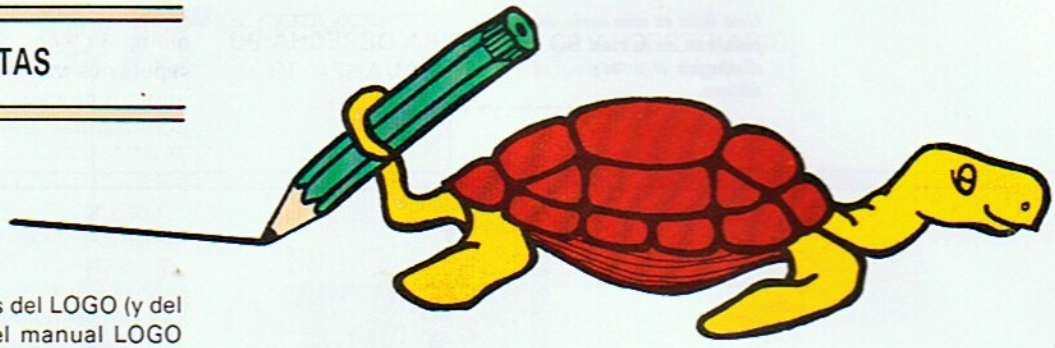
Uno de los inconvenientes que presenta el LOGO a pesar de su potencia es su lentitud.

Para finalizar esta sección dedicada a los gráficos vamos a dejar un procedimiento que realiza un dibujo complicado en comparación con lo que ocupa dicho procedimiento. Esto simplemente es una muestra de la originalidad a que puede dar lugar el LOGO con muy poco esfuerzo. ¿A qué no nos atreveríamos a hacer el siguiente programa en BASIC?

?PARA giros
>REPITE 4 (AVANZA 50 GIRADERECHA 90)
>GIRADERECHA 35
>giros
>FIN

Este procedimiento, como se puede observar, se llama a sí mismo incondicionalmente, con lo cual estará permanentemente ejecutándose a no ser que se impida con **BREAK**.

LAS PALABRAS Y LAS LISTAS



El LOGO está provisto de un lápiz con el cual va dejando su huella para hacer los dibujos.

Otro de los aspectos importantes del LOGO (y del cuál no se menciona nada en el manual LOGO de *LCSI/SOLI*) es el manejo de palabras y listas.

?ASIGNA ''paises (japón españa francia)

En este caso en la variable «paises» hemos introducido una lista formada toda ella de palabras. También podemos realizar una asignación a una variable de una lista formada a su vez por otra u otras listas además de palabras:

?ASIGNA ''paisesbis (europa (españa francia italia) japón)

Ahora en la variable «paisesbis» hay una lista formada por otra lista, la que se llama europa, que está a su vez formada por palabras.
Las listas podemos modificarlas, imprimirlas totalmente o imprimir sólo parte de ellas. Por ejemplo:

?IMPRIME PRIMERO :paises
japón
?IMPRIME ULTIMO :paises
francia

En el primer caso hemos tomado el primero de la lista dentro de la variable «paises»; y en el segundo nos aparecerá en pantalla el último de la lista. Pero, ¿qué pasa con los datos centrales?, ¿no podemos acceder a ellos? Ahora veremos cómo.

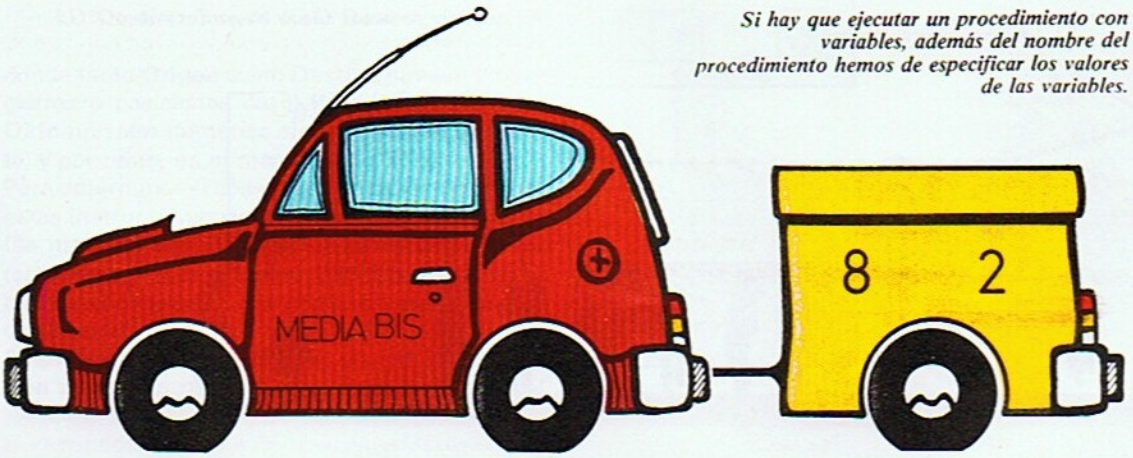
?IMPRIME MENOSPRIMERO :paises
españa francia
?IMPRIME MENOSULTIMO :paises
japón españa

Ya hemos accedido a la parte central de la lista, pero todavía seguímos teniendo pegado el primer y último elemento, de la lista, pero ya tenemos bastantes herramientas para coger cualquier palabra de nuestra lista.

?IMPRIME PRIMERO MENOSPRIMERO :paises
españa
?IMPRIME ULTIMO MENOSULTIMO :paises
españa

Lo que nos hace el primer caso es imprimir la primera palabra de la lista una vez quitado el primero, resumiendo, nos imprime el segundo. Igualmente, el siguiente caso nos imprime el último elemento una vez quitado el último, es decir, nos imprime el penúltimo igualmente, para seguir teniendo acceso a los datos más interiores de la lista basta con añadir tantos **''MENOSPRIMERO''** y **''MENOSULTIMO''** como sea necesario.
Podemos añadir también datos a una lista, no sólo extraerlos. Para ello utilizaremos los comandos **PONPRI** (abreviatura de PONer PRImero) y **PONULT** (abreviatura de PONer ULTimo).

Si hay que ejecutar un procedimiento con variables, además del nombre del procedimiento hemos de especificar los valores de las variables.



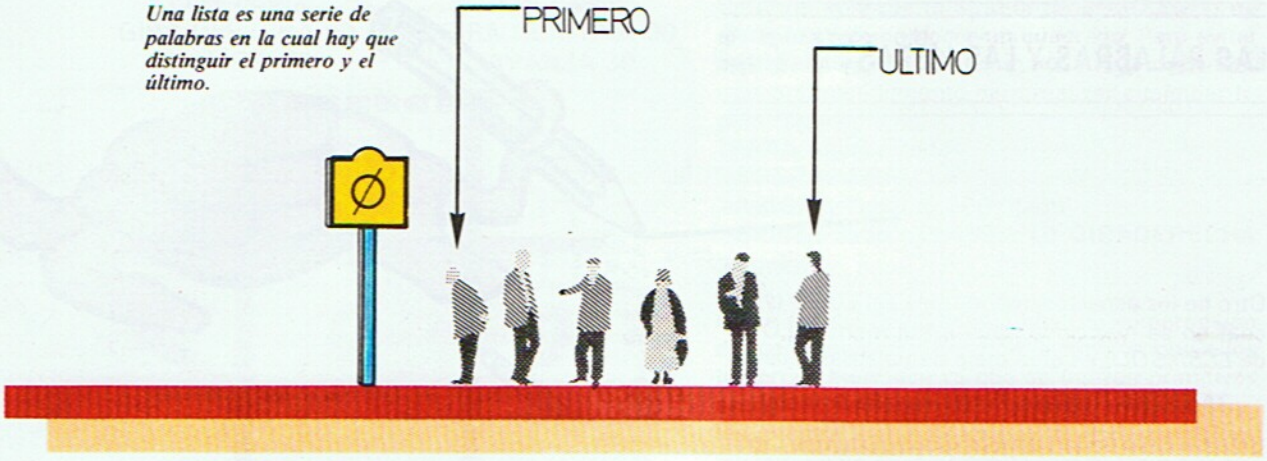
¡!

Uno de los incovenientes del lenguaje LOGO es su lentitud para hacer un programa largo y complicado.

*

El LOGO es un lenguaje que presenta algunas características novedosas sobresalientes que han sido incorporadas actualmente en otros lenguajes considerados como más potentes.

Una lista es una serie de palabras en la cual hay que distinguir el primero y el último.



?IMPRIME PONRI ''mexico :paises
mexico
?IMPRIME PONULT ''portugal :paises
portugal

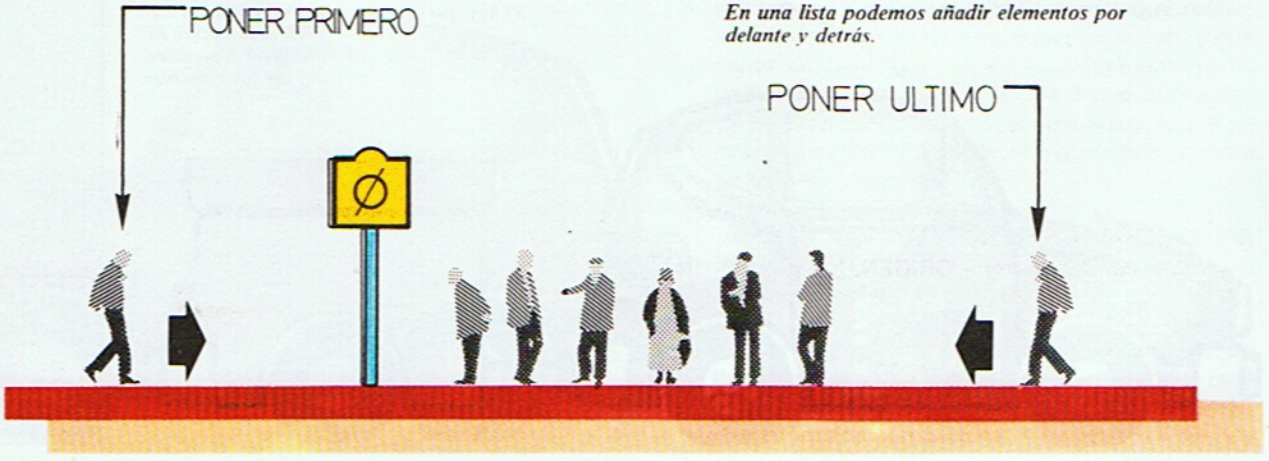
PARA FINALIZAR

Con el espacio limitado que ofrece este artículo no hay lugar para profundizar más en este lenguaje. Ahora es la originalidad del lector la que se pone a prueba para que ponga en práctica toda su habilidad para realizar programas. Se puede ayudar, para ello, el manual LOGO, que salvo algunas carencias (no menciona como trabajar con listas y palabras y no explica como utilizar el ZX MICRODRIVE, ya que se puede trabajar con él), es fácil de entender. Además, al final del mismo tiene un glosario de todas las palabras primitivas que utiliza y unas indicaciones de su sintaxis y de su función.

Creo que ha quedado claro que este lenguaje no puede ser considerado de segunda categoría aunque sea el más indicado para aprender. Aporta ventajas importantes, como el uso de los procedimientos, algo que el BASIC no presenta y que lo soluciona con los **GO TOs** y **GOSUBs** pero que en un programa largo complica la comprensión y el seguimiento del mismo. Será difícil que un iniciado atrape «malos vicios» con el LOGO y muy fácil que lo haga con el BASIC, que luego además serán traspasables a otros lenguajes con los problemas que ocasiona, sobre todo porque la mayor parte de los lenguajes informáticos utilizan procedimientos.

Uno de los inconvenientes del lenguaje LOGO es su lentitud para hacer un programa largo y complicado (se puede ganar algo de tiempo en los gráficos de tortuga escondiendo ésta). Sin embargo, el objetivo de este lenguaje no es su velocidad sino el abrir las puertas a aquellos que quieran adentrarse en el campo informático. Si es necesario realizar programas más complicados hay que utilizar otros lenguajes más potentes, como podría ser el PASCAL.

En una lista podemos añadir elementos por delante y detrás.



DE 8 EN 8 BITS

Sentadas ya las bases previas necesarias para la comprensión de los distintos grupos de instrucciones, comenzamos en este capítulo a realizar un análisis detallado de cada uno de ellos.
A parte de la misión concreta de una determinada instrucción (discutiremos siempre en el texto su funcionamiento), encontraremos otros datos de interés para la programación en C/M en las tablas que presentamos como desglose de cada grupo particular, cuyo significado y utilidad discutíamos en el bloque anterior, tales como los ciclos de reloj necesarios para su ejecución o el estado del registro de indicadores F.

REGISTRO	DECODIFICACION r ó r'
A	111
B	000
C	001
D	010
E	011
H	100
L	101

LD DESTINO, ORIGEN

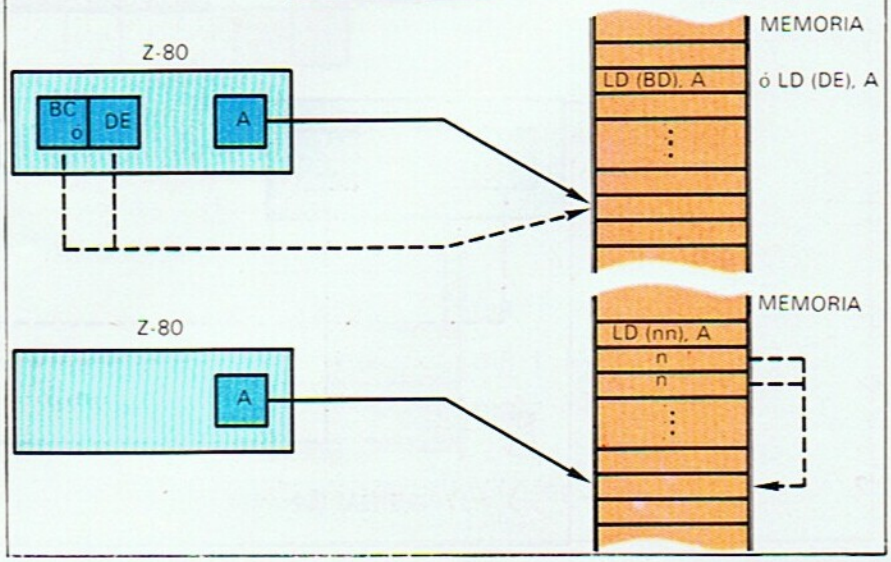
Volvamos a las tablas del capítulo 41. En ellas, bajo la denominación «Carga de ocho bits» encontramos una colección de instrucciones, todas ellas de código mnemónico LD (abreviatura del inglés *LoaD*, cargar) encargadas de tramitar los movimientos de información que involucren a registros, posiciones de memoria y datos individuales.
Todas ellas responden al formato

LD Destino,Origen o **LD Destino,Dato**

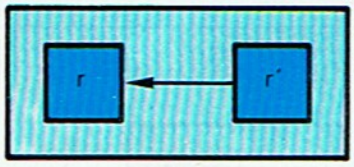
donde tanto **Origen** como **Destino** pueden ser registros o posiciones de memoria individuales y **Dato** un valor numérico almacenado en un octeto, y por tanto, en el rango de 0 a 255.
Para determinar el código de operación de una de estas instrucciones entraremos en la tabla por la fila marcada con DESTINO y buscaremos la intersección con la columna ORIGEN, lugar en el cual encontraremos los códigos hexadecimales correspondientes.
Si allí aparece un cuadro en blanco, la instrucción planteada no está contemplada dentro del juego presente en el microprocesador Z 80.
El resultado de una instrucción de «Carga» trae

ENSAMBLADOR	CODIGO MAQUINA	N.º BYTES	CICLOS RELOJ	CICLOS MAQUINA
LD (BC), A	0 0 0 0 0 0 1 0	1	7	2
LD (DE), A	0 0 0 1 0 0 1 0	1	7	2
LD (nn), A	0 0 1 1 0 0 1 0	3	13	4
(BAJO)	n n n n n n n n			
(ALTO)	n n n n n n n n			

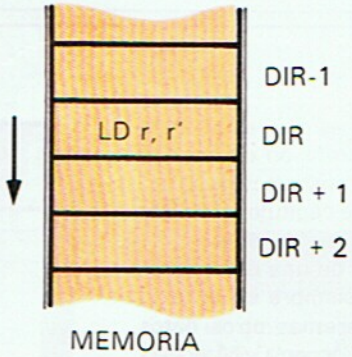
INDICADORES NO AFECTADOS



ENSAMBLADOR	CODIGO MAQUINA	Nº BYTES	CICLOS DE RELOJ	CICLOS MAQUINA
LD r, r'	0 1 r r r r' r' r'	1	4	1



Z-80



INDICADORES NO AFECTADOS

consigo la «copia» del contenido de la posición de memoria o registro Origen en el registro o posición de memoria **Destino**, manteniéndose SIEMPRE inalterado el primero por la ejecución de la instrucción (en su momento comentaremos una excepción, la cual confirma la regla).

Efectuando un recorrido por la tabla correspondiente es factible la siguiente clasificación:

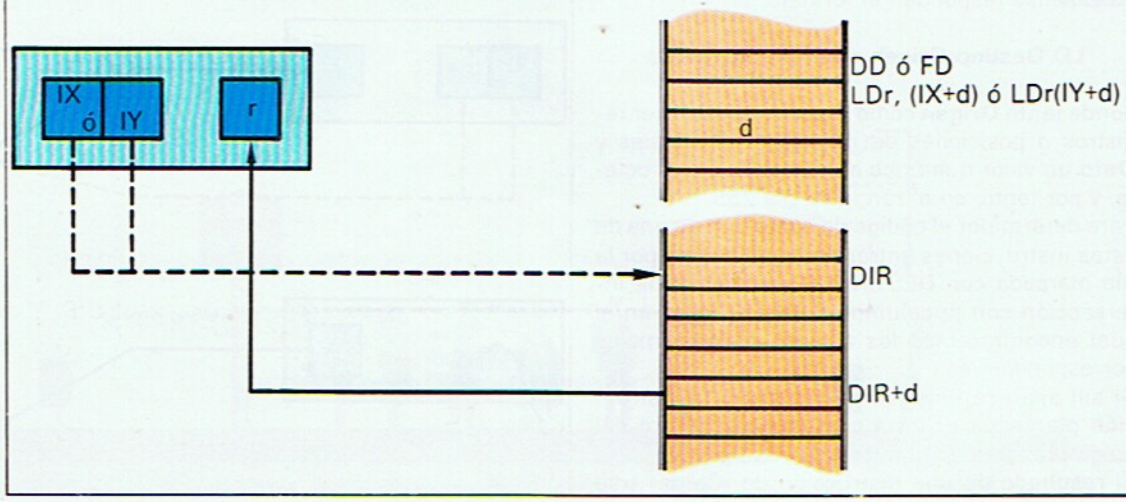
1) Carga en registros

1.1. Entre registros: **LD r,r'**

Con r y r' representamos cualquiera de los regis-

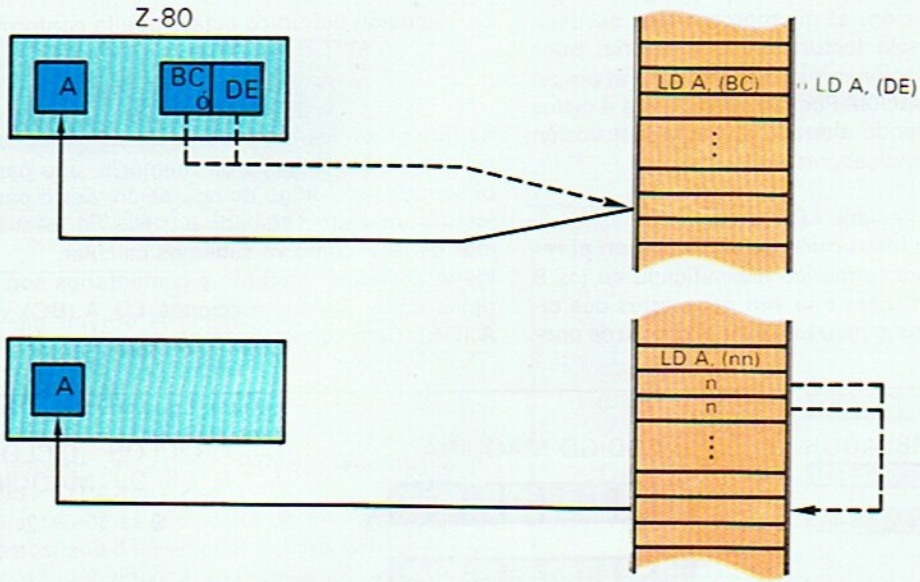
ENSAMBLADOR	CODIGO MAQUINA	Nº BYTES	CICLOS RELOJ	CICLOS MAQUINA
LD r, (IX+d)	1 1 0 1 1 1 0 1 0 1 r r r 1 1 0 d d d d d d d d	3	19	5
LD r, (IY+d)	1 1 1 1 1 1 0 1 0 1 r r r 1 1 0 d d d d d d d d	3	19	5

INDICADORES NO AFECTADOS



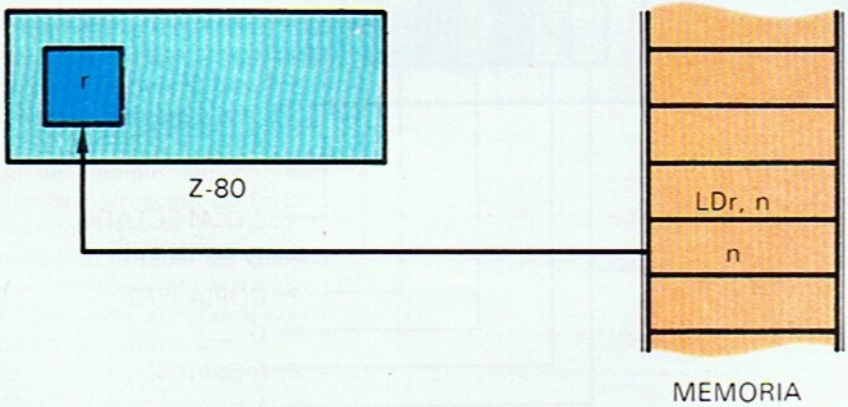
ENSAMBLADOR	CODIGO MAQUINA	N.º BYTES	CICLOS RELOJ	CICLOS MAQUINA
LD A, (BC)	0 0 0 0 1 0 1 0	1	7	2
LD A, (DE)	0 0 1 1 1 0 1 0	1	7	2
LD A, (nn)	0 0 1 1 1 0 1 0	3	13	4
(BAJO)	n n n n n n n n			
(ALTO)	n n n n n n n n			

INDICADORES NO AFECTADOS



ENSAMBLADOR LDr, n	CODIGO MAQUINA	N.º BYTES	CICLOS RELOJ	CICLOS MAQUINA
	0 0 r r r 1 1 0	2	7	2
	n n n n n n n n			

INDICADORES NO AFECTADOS



tros A, B, C, D, E, H o L y en la figura encontramos la decodificación seguida por el Z 80 para reconocerlos, siendo el formato general de esta instrucción en código objeto (código máquina).

01(-r-)(-r'-)

Consideremos, por ejemplo, la instrucción LD A,B perteneciente a este grupo. En las tablas encontramos su código de operación asociado: 78h. Utilizando la decodificación dada en la figura obtenemos el byte 01111000, el cual no es otra cosa que la representación binaria del código anterior. Para su ejecución, el microprocesador necesita realizar una sola lectura de la memoria, pues toda la información que precisa está en el propio código de operación. Por tanto, empleará 4 ciclos de reloj, no siendo alterado por esta instrucción el registro de indicadores.

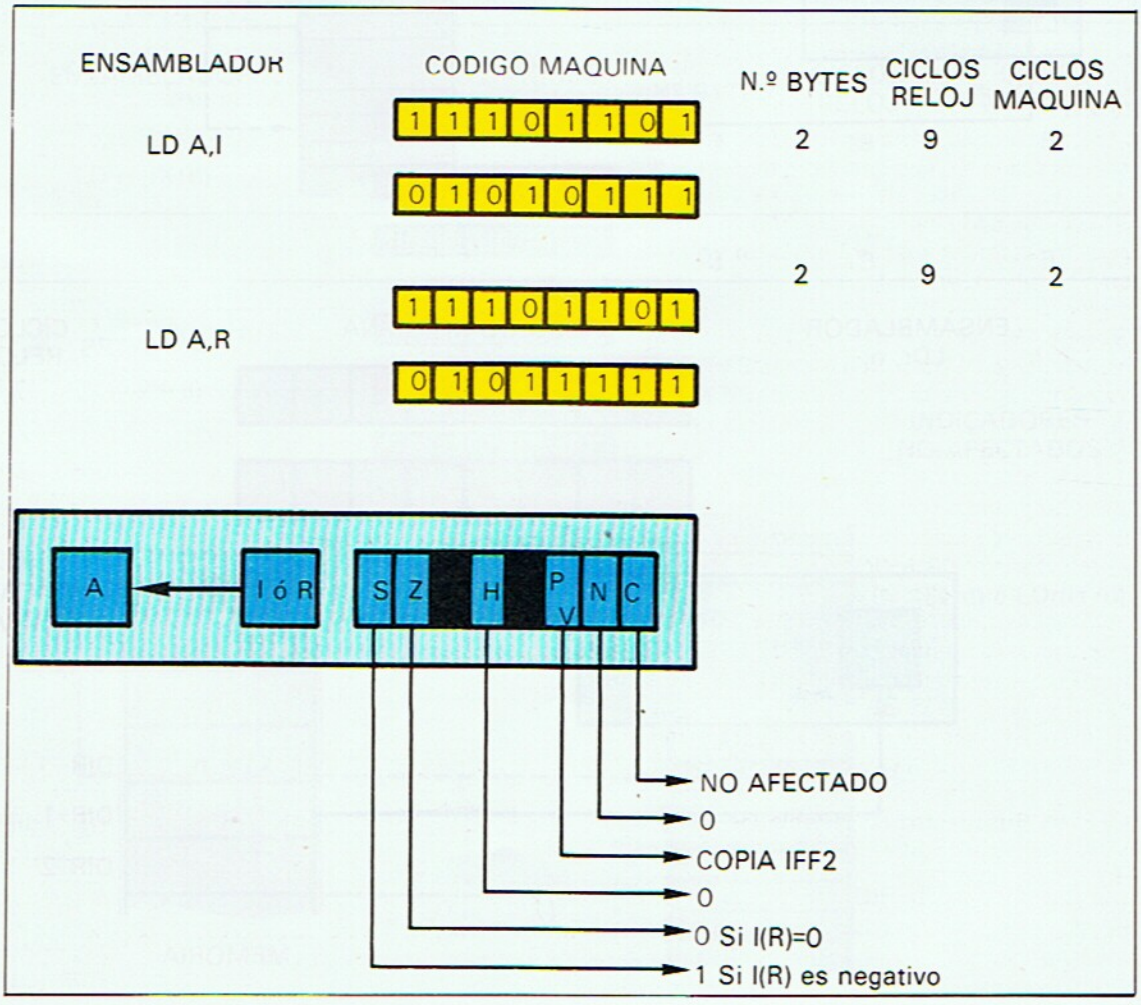
1.2. Carga inmediata: **LD r,n**
Mediante esta instrucción se almacena en el registro **r**, el valor numérico especificado en los 8 bits del dato **n**. Para ello son necesarios dos ciclos máquina: uno para localizar el código de operación (4 ciclos de reloj) y otro para leer el dato (3 ciclos) sin cambio en los indicadores de estado. Los dos bytes que la componen ofrecen el siguiente aspecto:

00(-r-)110
(---n---)

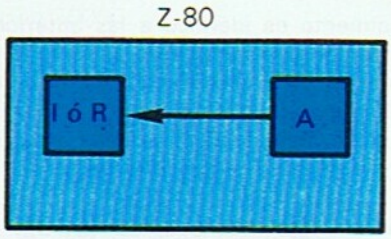
1.3. Carga indirecta: **LD r,(HL)**
Estas instrucciones recurren al par de registros HL como puntero de una determinada posición de memoria, donde leer el byte a ser copiado en el registro **r**.
La ejecución del único octeto que la conforma

01(-r-)110

no influye en los indicadores de condición, siendo necesarios 2 ciclos de memoria, uno para la búsqueda del código de operación y otro para la lectura del dato señalado por HL. En total 7 ciclos de reloj como ya sabemos calcular.
Idéntico comportamiento y comentarios son factibles sobre las instrucciones **LD A,(BC)** y **LD A,(DE)**. Como observaremos, en ellas **r** debe ser



ENSAMBLADOR	CODIGO MAQUINA	N.º BYTES	CICLOS RELOJ	CICLOS MAQUINA
LD I,A	1 1 1 0 1 1 0 1 0 1 0 0 0 1 1 1	2	9	2
LD R,A	1 1 1 0 1 1 0 1 0 1 0 0 1 1 1 1	2	9	2



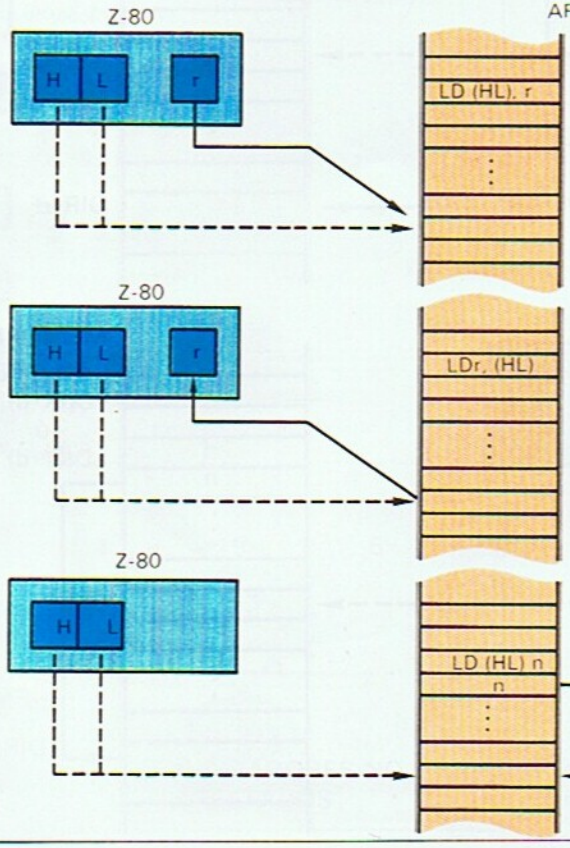
INDICADORES:
NO AFECTADOS

ineludiblemente el acumulador A, pues de otra manera, no están contempladas en el juego del Z 80.
En este mismo grupo podríamos incluir una instrucción, la cual exclusivamente trabaja sobre el acumulador sirviéndose de direccionamiento extendido: **LD A,(nn)**.
Mediante ella se almacena en éste el contenido de la posición de memoria definida por los dos octetos nn, debiéndose especificar en primer lugar el byte bajo de tal dirección (recordemos que cuando aparecen los paréntesis en una instrucción escrita en ensamblador debe leerse «el contenido de...»).
Emplea 13 ciclos de reloj repartidos, 4 para buscar el código de operación, 3 más 3, para leer los dos octetos nn, y finalmente otros 3 al buscar el dato contenido en la posición nn. Los indicadores no se ven afectados.

1.4. Carga Indexada: **LD r,(IX+d)** y **LD r,(IY+d)**
El objetivo de estas instrucciones es colocar en el reigstro **r** el contenido de la posición de memoria obtenida al sumar el valor actual del par de registros indice IX o IY (según empleemos la primera o la segunda) y un byte de desplazamiento **d** expresado en complemento a dos, y por tanto, en el rango –128 a +127.
Aclaremoslo con un ejemplo: supongamos que deseamos ejecutar la instrucción **LD B,(IX-10)** y que en ese momento el par IX contiene 32767 (7FFFh). Busquemos en la tabla los códigos hexadecimales asociados donde encontraremos **DD 46 d**. Nos queda, por tanto, sustituir **d** por su valor en dicho sistema de numeración.
Pues su valor es F6. ¿Por qué? Recordemos que el desplazamiento hemos de expresarlo en complemento a dos, y precisamente F6 es la repre-

ENSAMBLADOR	CODIGO MAQUINA	N.º BYTES	CICLOS RELOJ	CICLOS MAQUINA
LD (HL), r	0 1 1 1 0 r r r	1	7	2
LDr, (HL)	0 1 r r r 1 1 0	1 2	7 10	2 3
LD (HL), n	0 0 1 1 0 1 1 0 n n n n n n n n			

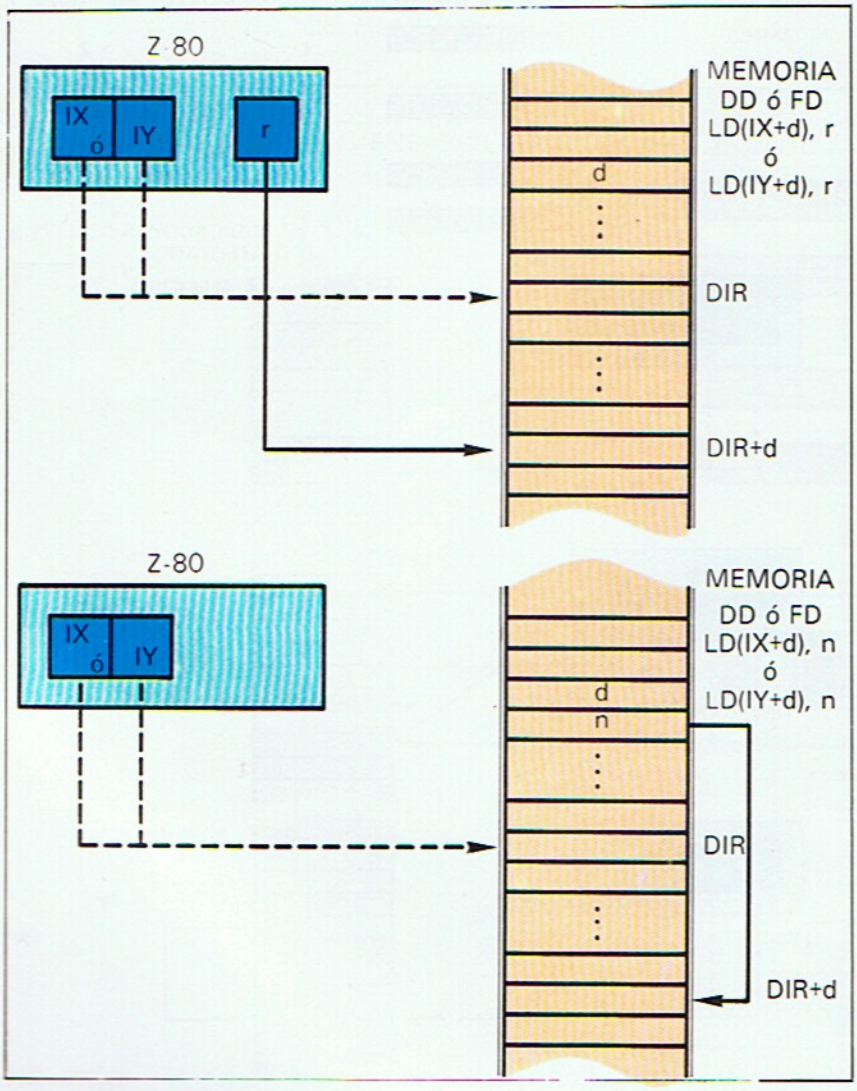
INDICADORES NO AFECTADOS



sentación en este sistema del número decimal -10. Con ésto la instrucción completa es **DD 46 F6**. ¿Qué ocurre al ejecutarla?
El microprocesador buscará el valor del par de registros IX (32767 en el ejemplo). A continuación, restará 10 unidades de éste obteniendo 32757. Finalmente, extraerá los ocho bits almacenados en esta posición de memoria y los cargará en el registro B.
Podemos comprobar en las tablas que el primer byte de todas las instrucciones que utilizan el registro IX es DD, y el de las que manejan el IY, FD.

2) Carga en memoria
En cierto modo, las instrucciones del presente grupo pueden considerarse como inversas de las anteriores, al menos si tenemos en cuenta que el papel desempeñado por los registros o posiciones de memoria **Origen** y **Destino** afectados se intercambia respecto al sentido en que se efectuaba la transferencia de datos.

2.1. Carga indirecta: **LD (HL),r** y **LD (HL),n**



Al ejecutar estas instrucciones se consigue almacenar en el octeto de memoria especificado por el par de registros HL, el valor actual del registro **r** o del dato **n**. Volvemos a insistir en que tras el proceso, tanto HL, como **r** o **n** se mantienen inalterados.
Dentro de este mismo apartado es posible considerar tres instrucciones las cuales sólo tienen sentido cuando el registro **Origen** es el acumuldor A:

LD (BC),A LD (DE),A LD (nn),A

El funcionamiento es idéntico a las anteriores, pero el contenido de A es transferido a la dirección de memoria señalada por BC, DE o los dos bytes **nn**. Recuerda que al utilizar la última, los octetos **nn** deberán especificarse en el orden bajo-alto.

2.2. Carga indirecta indexada: **LD (IX+d),r LD (IX+d),n LD (IY+d),r LD (IY+d),n**
Para establecer la posición de memoria donde almacenar el contenido del registro **r** o el dato **n**, el microprocesador sigue el mismo sistema que explicábamos cuando nos referíamos en el grupo 1.4 a las instrucciones que manejan direccionamiento indexado.

3) Carga sobre registros especiales. **LD A,I LD A,R LD I,A LD R,A**
Manejando las dos primeras conseguimos trasladar al acumulador el contenido del vector de página de interrupción I, o el valor del registro de regeneración de memoria R, respectivamente. Son las únicas instrucciones dentro del grupo de carga de 8 bits que alteran el contenido del registro de indicadores. Veamos cuál es el efecto sobre ellos:
C No es afectado por estas instrucciones.
Z Se coloca a 0 si I (R) vale 0.
S Encontraremos un 1 cuando el bit más significativo de I (R) sea 1.
N Se pone a 0.
H Contendrá 0.
P/V Copia el estado de IFF2, es decir, 0 cuando las interrupciones están inhibidas y 1, cuando están habilitadas.
Muy bien, ¿pero qué quiere decir IFF? Entre los múltiples circuitos presentes en el Z 80 dos son los encargados de señalarle cuando están habilitadas las interrupciones. Son denominados IFF1, el cual las permite o no, e IFF2 que sirve de almacenamiento temporal del estado de IFF1. Las siglas IFF corresponden según su denominación inglesa a *Interruption enable Flip-Flop* y de ahí que en castellano normalmente se conozca como «báscula de habilitación de interrupciones».
Cuando ejecutamos una instrucción EI (habilitar interrupción), IFF1 e IFF2 son puestos al unísono a 1. Al ser tratada ésta, simultáneamente pasan

a ser 0 y deberemos colocarlos por programa, nuevamente a 1, si queremos tratar otra interrupción. Todo esto será comentado con detalle en el capítulo dedicado a las interrupciones.
Otra característica a tener en cuenta se refiere a la periódica variación a que está sujeto el registro R (recordemos la excepción comentada anteriormente). Esta circunstancia puede aprovecharse en los programas en que necesitemos generar números aleatorios.
Si queremos comprobarlo introduzcamos en el cargador hexadecimal la siguiente línea y ejecutemos el programa ubicándolo a partir de la dirección 30000:

```
10 DATA "ED 5F 4F 06 00 C9 **",612
```

Ahora sustituyamos la línea 10 por la siguiente:

```
10 PRINT USR 30000: PAUSE 0: GOTO 10
```

y volvamos a ejecutar **RUN**. Cada vez que pulsemos una tecla obtendremos en la pantalla el valor almacenado en R, al ejecutarse la rutina en C/M. Observaremos que éstos siempre están comprendidos entre 0 y 127, es decir, el bit de mayor peso del registro R siempre es 0.
Las instrucciones **LD I,A** y **LD R,A** introducen en los registros I o R, respectivamente, el contenido actual del acumulador sirviéndose de direccionamiento implícito. Con ésto completamos el recorrido efectuado a través del grupo de carga de 8 bits.

ENSAMBLADOR	CODIGO MAQUINA	N.º BYTES	CICLOS RELOJ	CICLOS MAQUINA
LD(IX+d), r	1 1 0 1 1 1 0 1	3	19	5
	0 1 1 1 0 r r r			
	d d d d d d d d			
LD(IY+d), r	1 1 1 1 1 1 0 1	3	19	5
	0 1 1 1 0 r r r			
	d d d d d d d d			
LD(IX+d), n	1 1 0 1 1 1 0 1	4	19	5
	0 0 1 1 0 1 1 0			
	d d d d d d d d			
	n n n n n n n n			
LD(IY+d), n	1 1 1 1 1 1 0 1	4	19	5
	0 0 1 1 0 1 1 0			
	d d d d d d d d			
	n n n n n n n n		INDICADORES NO AFECTADOS	

MORSE

Con MORSE podremos codificar todos los mensajes secretos que queramos y repetirlos hasta la saciedad. Es una buena forma de divertirse con un programa diferente.

EL PROGRAMA

Ante todo, es importante señalar que este programa utiliza unas llamadas a una subrutina en código máquina por lo que, antes de ejecutarlo, tras introducirlo en el Spectrum, es conveniente grabar el BASIC, con lo que evitaremos males mayores en descuidos inoportunos. Esta subrutina la hemos utilizado en anteriores programas para dar una mayor vistosidad a las pantallas. Se trata de la subrutina de caracteres gigantes de PSION COMPUTERS.

Relativo al uso de la citada rutina, hemos incluido en el programa que nos ocupa una ligera variación: la subrutina de caracteres gigantes trabaja con dos tipos diferentes de generadores de caracteres. El primero de ellos es el propio del ordenador almacenado en R.O.M. a partir de la dirección 15616. El segundo generador está almacenado en R.A.M. a continuación de 54016. Para poder acceder a este segundo generador, el Spectrum efectúa dos **POKE** a la Variable del Sistema CHARS (direcciones 23606 y 23607) para que el puntero de dicha variable señale la nueva zona del generador de caracteres (256 bytes antes).

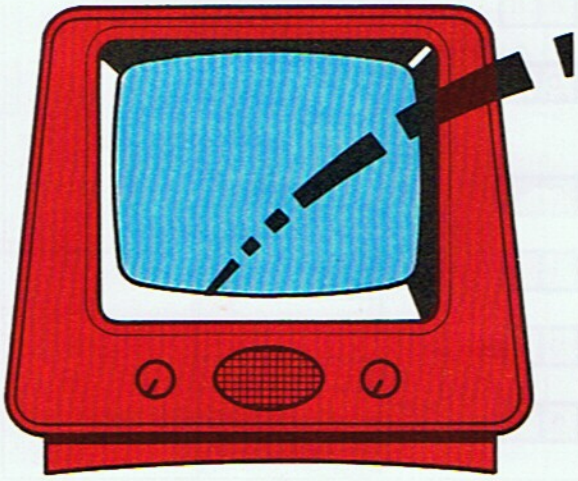
Es importante saber que los mensajes deben ser siempre introducidos en minúsculas.

Nada más ejecutar el programa, aparecerá en pantalla un menú con dos opciones: 1.º conversión alfabeto-morse y 2.º fin. Lógicamente pulsaremos la tecla 1 para ejercitarnos con el programa.

Es importante saber que los mensajes deben ser siempre introducidos en minúsculas. De entrada, el programa selecciona para el cursor el modo de minúsculas. No hay que alterarlo bajo ninguna circunstancia.

Una vez introducido el mensaje, debemos esperar unos pocos segundos para ver la nueva codificación a alfabeto Morse. El tiempo de espera está en función de la longitud del texto. Cuanto más largo sea, más extensa será nuestra espera. Cuando el Spectrum haya terminado la codificación, irán apareciendo las sucesiones de puntos

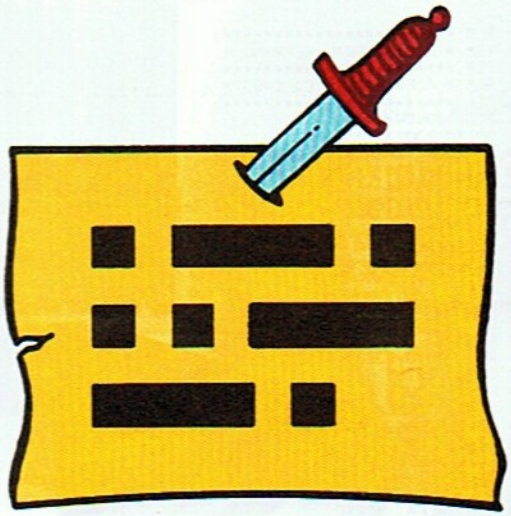
Cuando el Spectrum haya finalizado la codificación irán apareciendo las sucesiones de puntos y rayas que forman el mensaje.



y rayas, a la vez que entra en funcionamiento el altavoz del ordenador simultaneando con tonos cortos (puntos) y largos (rayas) la representación de pantalla.

ADOPCION DEL PROGRAMA

Para la adopción del programa hemos de proceder en primer lugar a la grabación del soporte BASIC que se lista en el presente capítulo, grabando a continuación en la cinta dos bloques de memoria: el correspondiente al código máquina de la subrutina de caracteres gigantes y el nuevo generador de caracteres tipo DATA que apareció lis-



Con MORSE podremos codificar todos los mensajes secretos que queramos y repetirlos hasta la saciedad.

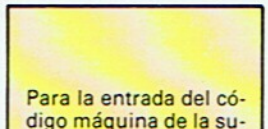
tado en el programa IMPRENTA. Es indiferente el orden en que efectuemos la grabación de estos dos últimos bloques de memoria.

Para la entrada del código máquina de la subrutina de caracteres gigantes utilizaremos el mismo sistema que ya hemos seguido en otras ocasiones, cargándola a partir de la dirección 32256.

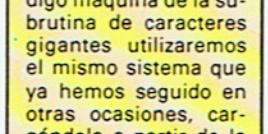
Para el nuevo generador de caracteres, utilizaremos el procedimiento explicado en el programa IMPRENTA, otorgando como dirección de carga del mismo 54256.

Es importante grabar el listado BASIC con autoejecución en la línea 190: **SAVE "MORSE" LINE 190.**

Para la entrada del código máquina de la subrutina de caracteres gigantes utilizaremos el mismo sistema que ya hemos seguido en otras ocasiones, cargándola a partir de la dirección 32256.



Para el nuevo generador de caracteres, utilizaremos el procedimiento explicado en el programa IMPRENTA, otorgando como dirección de carga del mismo 54256.



La ausencia de caracteres subrayados en el listado se debe a que el programa MORSE no utiliza ningún gráfico UDG.

```
  10 REM ************************
  20 REM * J.M.MAYORAL SERRANO *
  30 REM ************************
  40 REM * ALFA.MORSE (C) 1985 *
  50 REM ************************
  60 POKE 23658,0
  70 GO TO 250
  80 REM SBR. CARACT. GIGANTES
  90 LET xx=(256-8*xs*LEN p$)/2
 100 LET i=23306
 110 POKE i,xx: POKE i+1,yy: POKE i+2,xs: POKE i+3,ys
: POKE i+4,8
 120 LET i=i+4: LET w=LEN p$
 130 FOR q=1 TO w
 140 POKE i+q,CODE p$(q)
 150 NEXT q
 160 POKE i+w+1,255
 170 LET w=USR 32256
 180 RETURN
 190 REM  AUTO-EJEC. OBLIGATORIA
 200 CLEAR 32255
 210 LOAD ""CODE
 220 LOAD ""CODE
 230 POKE 23606,240
 240 POKE 23607,210
 245 POKE 23658,0
 250 RESTORE : LET PUNTO=.1: LET LINEA=.3: LET PAUSA=
30
 260 REM  ALFA. MORSE
 270 DATA ".-.-."
 280 DATA ".-.-.-"
 290 DATA "-..-.."
 300 DATA "......"
 310 DATA "-..-."
 320 DATA "-----"
 330 DATA ".----"
 340 DATA "..---"
 350 DATA "...--"
 360 DATA "....-"
 370 DATA "....."
 380 DATA "-...."
 390 DATA "--..."
 400 DATA "---.."
 410 DATA "----."
 420 DATA "---..."
 430 DATA "-.-.-."
 440 DATA ".-"
 450 DATA "-..."
 460 DATA "-.-."
 470 DATA "-.."
 480 DATA "."
 490 DATA "..-."
 500 DATA "--."
 510 DATA "...."
 520 DATA ".."
 530 DATA ".---"
 540 DATA "-.-"
 550 DATA ".-.."
 560 DATA "--"
 570 DATA "-."
 580 DATA "---"
 590 DATA ".--."
 600 DATA "--.-"
 610 DATA ".-."
 620 DATA "..."
 630 DATA "-"
 640 DATA "..-"
 650 DATA "...-"
 660 DATA ".--"
 670 DATA "-..-"
 680 DATA "-.--"
 690 DATA "--.."
 700 DATA "..--.."
 710 DATA "-...-"
 720 DATA ".----."
 730 DATA "--..--"
 740 DATA ".-.-"
 750 DATA "..-.."
 760 DATA "---."
 770 DATA ".--"
 780 DIM C$(4,5): DIM N$(17,6): DIM A$(26,4)
 790 FOR K=1 TO 17: READ N$(K): NEXT K
 800 FOR K=1 TO 26: READ A$(K): NEXT K
 810 READ D$,U$,O$,E$
 820 FOR K=1 TO 4: READ C$(K): NEXT K:
 830 BORDER 0: PAPER 0: CLS
 840 BRIGHT 1: LET P$=" Prog. MORSE ": LET YY=50: LET
 XS=2: LET YS=3: INK 0: PAPER 2: GO SUB 1250
 850 BRIGHT 0: PAPER 0: INK 6: LET YY=100: LET XX=30:
 LET XS=1: LET YS=2: LET P$="1    ALFABETO - MORSE.":
 GO SUB 1260
 860 INK 4: LET YY=120: LET P$="2    SALIDA DE PROGRA
MA.": GO SUB 1260
 870 PRINT  OVER 1;AT 21,4;"PULSA LA OPCION DESEADA"
 880 LET K$=INKEY$
 890 IF CODE K$<49 OR CODE K$>50 THEN  BEEP .1,10: GO
 TO 870
 900 IF VAL K$=2 THEN  GO TO 10000
 910 BORDER 1: PAPER 1: CLS
 920 LET P$="INTRODUCE  EL": LET YY=100: LET XS=2: IN
K 5: GO SUB 1250
 930 LET P$="MENSAJE": LET YY=125: LET XS=3: INK 6: G
O SUB 1250
 940 POKE 23606,0
 950 POKE 23607,60
 960 INPUT  LINE M$: CLS : IF LEN m$>119 THEN  GO TO
1310
 970 POKE 23606,240
 980 POKE 23607,210
 990 LET P$="MORSE": LET YY=120: LET YS=7: LET XS=6:
PAPER 7: INK 1: GO SUB 1250: PAPER 1: INK 9
1000 DIM J(LEN M$): DIM H$(LEN M$,6): LET X=0: FOR K=
1 TO LEN M$
1010 LET K$=M$(K)
1020 IF K$=" " THEN  LET J(X)=PAUSA
1030 IF X>0 AND K$="H" THEN  IF H$(X)="-.-. " THEN  L
ET H$(X)="----": GO TO 250
1040 IF K$="´" THEN  IF M$(K-1)<>" " THEN  GO SUB 127
0
1050 IF K$="´" AND K>1 THEN  IF M$(K-1)=" " THEN  LET
 X=X+1: LET H$(X)=O$
1060 IF CODE K$>=43 AND CODE K$<=59 THEN  LET X=X+1:
LET H$(X)=N$(CODE K$-42)
1070 IF CODE K$>=97 AND CODE K$<=122 THEN  LET X=X+1:
 LET H$(X)=A$(CODE K$-96)
1080 IF K$="?" THEN  LET X=X+1: LET H$(X)=D$
1090 IF K$="=" THEN  LET X=X+1: LET H$(X)=U$
1100 IF K$="!" THEN  LET X=X+1: LET H$(X)=E$
1110 NEXT K
1120 FOR K=1 TO X
1130 PRINT H$(K);"  ";
1140 FOR Z=1 TO 6
1150 IF H$(K,Z)="." THEN  BEEP PUNTO,40
1160 IF H$(K,Z)="-" THEN  BEEP LINEA,40
1170 NEXT Z
1180 PAUSE 6+J(K)
1190 NEXT K
1200 GO SUB 1270: LET XS=1: LET YS=3: LET P$="QUIERES
 QUE LO REPITA": LET YY=140: INK 7: GO SUB 1250
1210 PAUSE 0
1220 IF INKEY$="s" THEN  CLS : LET P$="M O R S E": LE
T YS=INT (RND*4)+2 : LET XS=INT (RND*2)+2: INK INT (R
ND*3)+5: GO SUB 1250: INK 9: GO TO 1120
1230 PAPER 0: BORDER 0: CLS : GO TO 830
1240 CLS
1250 GO TO 90
1260 GO TO 100
1270 FOR W=14 TO 21
1280 PRINT AT W,0; PAPER 1;"
                                    "
1290 NEXT W
1300 RETURN
1310 CLS
1320 LET P$="EL MENSAJE ES DEMASIADO LARGO"
1330 INK 4
1340 LET XS=1: LET YS=2: LET YY=70: GO SUB 1250
1350 PAPER 1: INK 5
1360 LET P$="EL MENSAJE NO DEBE SUPERAR"
1370 LET YS=2: LET YY=100
1380 GO SUB 1250
1390 LET P$="LOS 119  CARACTERES"
1400 LET YY=120:
1410 GO SUB 1250
1420  PRINT  INK (INT (RND*6)+2);AT 21,1;"Pulsa una t
ecla para continuar"
1430 LET k$=INKEY$
1440 IF k$="" THEN  GO TO 1420
1450 RESTORE
1460 CLS
1470 POKE 23606,240
1480 POKE 23607,210
1490 GO TO 780
```

FORTH

ACTUALMENTE el lenguaje informático más popular entre los microordenadores es el BASIC, debido a la poca memoria que ocupa y a su relativa facilidad de aprenderlo. Sin embargo, presenta el inconveniente de que al ser «interpretado» se hace excesivamente lento para algunas aplicaciones, como los juegos. Se solucionaría el problema utilizando entonces un lenguaje compilado, pero son más difíciles de dominar y lentos para realizar continuas transformaciones.

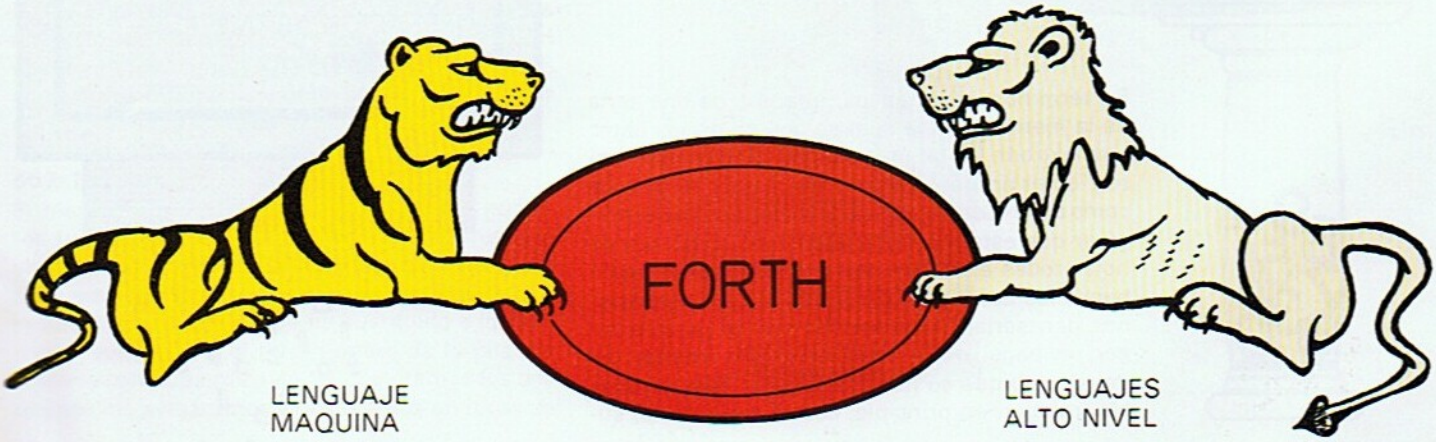
Es evidente que la solución ideal se encuentra en reunir las ventajas de ambos. Y esto es lo que ha conseguido el FORTH: un intérprete muy rápido. Este asombroso prodigio se consigue porque el FORTH trabaja de forma similar a como le gusta al ordenador, cercano al código máquina. Así, por ejemplo, la forma en que guarda la información es en una pila o *stack* de la cual se extraen los datos para operar. Nos podemos imaginar que utiliza una columna en la cual se van apilando los números y los extraemos en orden inverso al de su entrada, como más tarde veremos.

Sin embargo, no todo puede ser perfecto. El FORTH presenta la particularidad o inconveniente de que no puede trabajar con decimales ni «sabe» realizar más operaciones matemáticas que las cuatro básicas (suma, resta, multiplicación y división), lo cual le hace inútil para una gran cantidad de aplicaciones. Entonces ¿cuál es la utilidad de este lenguaje? Sencillamente los juegos (para llevar un marcador no hace falta decimales). ¿Un lenguaje casi diez veces más rápido que el BASIC, intérprete y de alto nivel para hacer juegos? No está mal, ¿verdad?

La ventaja que aporta el FORTH con respecto al BASIC es la de su velocidad, pero su forma de trabajo es complicada.

¿Por qué no conocemos ya a este lenguaje? Con el SPECTRUM podemos, ya que disponemos de al menos dos versiones FORTH; la de *ARTIC*, co-

Su alta velocidad, a pesar de ser intérprete es debido a que se encuentra cercano al lenguaje máquina, pero sin abandonar las características de los lenguajes de alto nivel.

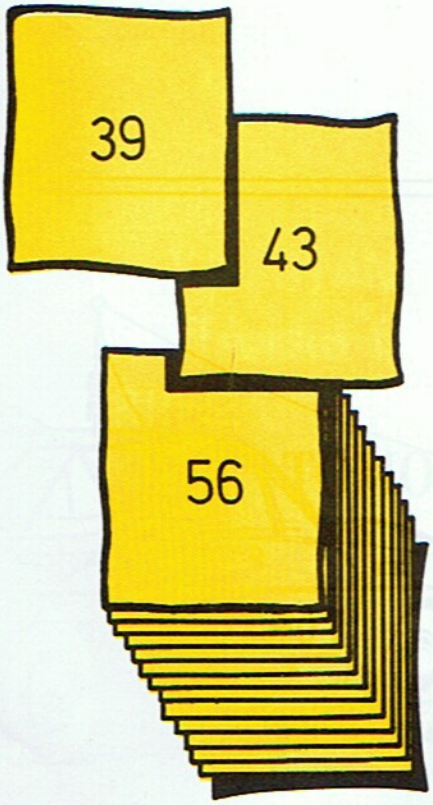


¡!

La única posibilidad que tiene el FORTH de operar es la de tomar los datos de la pila, si no es así no puede realizar operaciones ni funciones de ningún tipo.

*

Una pila o *stack* es una zona de la memoria en la cual se acumulan los datos y se extraen de tal manera que el último en entrar es el primero en salir.

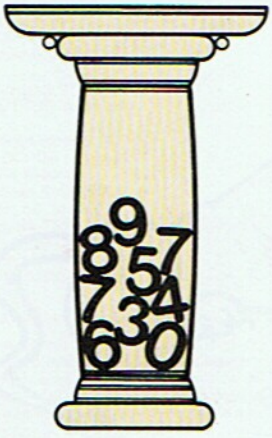


La pila la podemos comparar a unas hojas de papel amontonadas de las cuales sólo podemos trabajar con las de arriba para que así no se nos caigan las hojas.

mercializada por INDESCOMP, y la de *ABERSOFT.*

PILAS O *STACKS*

El FORTH guarda los datos en una columna que se conoce como pila o stack.



En términos informáticos, una pila es una zona de la memoria en la cual se acumulan los datos y se extraen de tal manera que el último en entrar es el primero en salir (en inglés se conoce como *LIFO «Last In, First Out»).* Nos podemos imaginar que estamos trabajando con un montón de hojas todas ellas apiladas y en las cuales escribimos los datos para luego realizar las operaciones necesarias con ellos. No nos es posible coger un papel de la parte central o inferior del montón porque se nos caería todo. Sólo es posible trabajar, en principio, con el papel (dato) superior, o, como mucho, con los tres superiores ya que con éstos no hay riesgo de accidente. Y ¿ésto que tiene que ver con FORTH? Pues nada más ni menos que ésta es la manera en la que trabaja este lenguaje.

Esto nos obliga a que los datos con los cuales vamos a trabajar estén dentro del *stack,* es decir, para operar debemos tener nuestros papeles con los datos escritos en nuestro montón. Veamos un ejemplo:

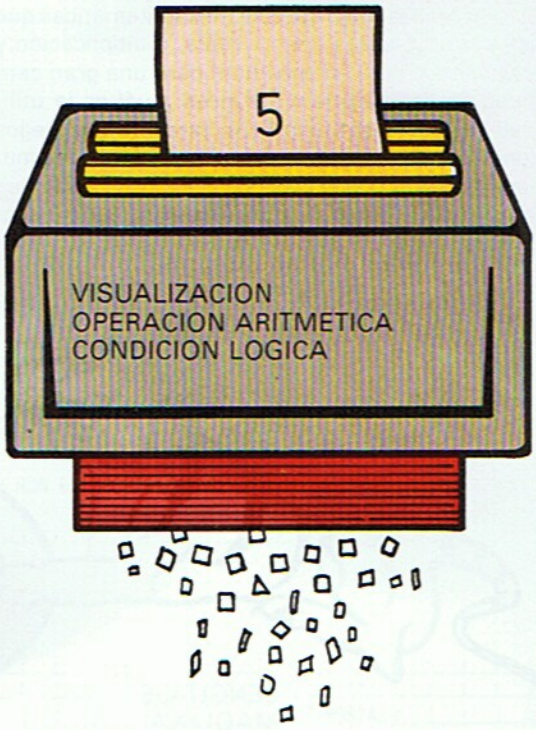
3.

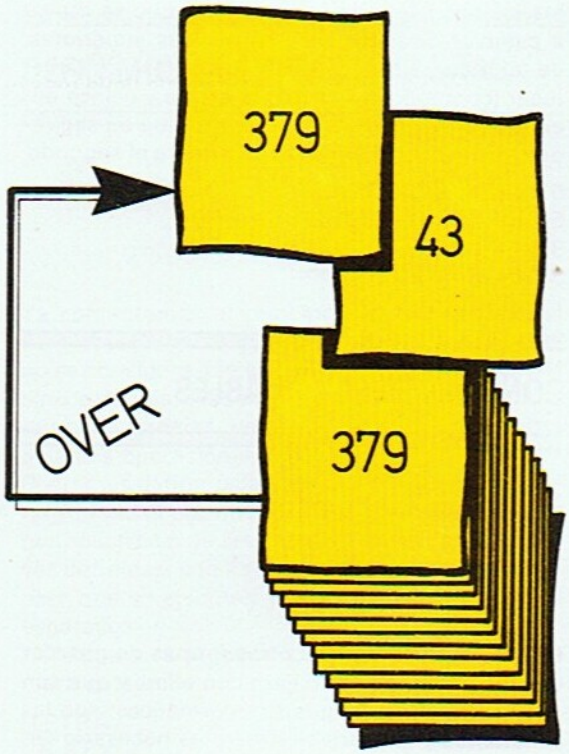
Sencillo ¿verdad? Vamos a ver lo que hace. En primer lugar introducimos un número en la pila (el 3) y después hacemos que se imprima el último número de la pila, que, evidentemente será el 3, gracias al comando "." (en FORTH a los comandos se les llama palabras). Siempre después de realizada una operación con un número de la pila (en este caso la visualización) éste es eliminado de ella. En nuestro ejemplo con el montón de papeles es como si después de ver el número del papel superior le rompiésemos, con lo cual tenemos acceso al siguiente inferior.

456 238 CR.
238 OK

Si hemos entendido el anterior ejemplo, éste nos costará poco. Tras introducir dos números en la pila visualizamos uno de ellos en la línea siguiente (para ello tenemos **CR**). Como el último intro-

Después que se realiza una operación con el número superior de la pila éste es destruido.





OVER *repetirá el valor del penúltimo dato del stack y lo pondrá encima de todos.*

ducido es el 238, éste será el que aparezca. En nuestro ejemplo de los papeles, primero escribimos uno con el 456 y encima ponemos otro con el 238. A la hora de imprimir será éste el que aparezca en primer lugar. Si ahora hacemos:

CR.

Es lógico que aparezca:

456 OK

Al imprimir, y por tanto destruir, el papel con el 238 se quedó al descubierto el siguiente inferior (el 456) que es el que ahora aparece por pantalla. Es decir, los últimos datos en entrar son los primeros en salir.

Hasta aquí todo va bien, pero ¿dónde están las operaciones? Ya mencionamos antes que FORTH opera con los números de la pila, concretamente con los dos últimos, y ambos son sustituidos por el resultado. La sintaxis es:

132 434 218 + CR..

652 132 OK

Primeramente introducimos una serie de números (un detalle: en FORTH los separadores son siempre los espacios) en la pila y ordenamos que se sumen los superiores. Por fin, nos imprime el resultado de la suma (que ha sustituido a los números sumados) y el otro número de la pila. Es como si con nuestros papeles cogiéramos los dos últimos los sumáramos y pusiéramos en lugar de ambos el resultado. De manera similar se hace con las demás operaciones: resta (–), multiplicación (*) y división (/).

Sin embargo, nuestro montón de papeles no es algo rígido e inmutable, sino que con cuidado podemos realizar algunos cambios, lo cual nos dará mayor facilidad de cálculo.

48 23 11 DUP * CR.

121 OK

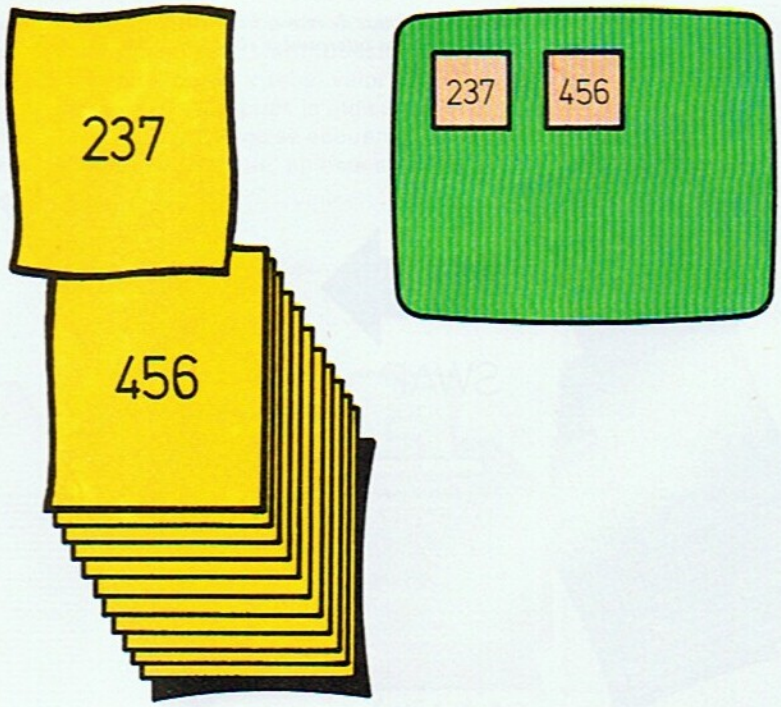
Aquí tenemos una palabra (recordemos que palabra es equivalente a comando u orden en FORTH) que nos permite realizar unos cambios en nuestra pila. En primer lugar introducimos tres números y al último de ellos le duplicamos (**DUP**), con lo cual ahora tendremos almacenados los números **48 23 11 11**. Por último se multiplican los dos números superiores y se sustituyen por el resultado, que es impreso obteniendo así el cuadrado de ese número. Con **DUP** es como si sacásemos una copia del papel que se encuentra arriba y lo depositásemos encima de todos.

También podemos coger el número superior y eliminarle sin hacer ninguna operación (lo contrario a **DUP**). Para ello utilizamos **DROP**. Con **OVER** sacamos una copia del penúltimo número de la pila y lo coloca encima de todos. Vamos a combinar las dos palabras para comprobar en la práctica su uso.

35 43 12 OVER DROP CR...

12 43 35 OK

Los datos se van almacenando unos encima de los otros y se extraen en orden inverso al de su introducción.

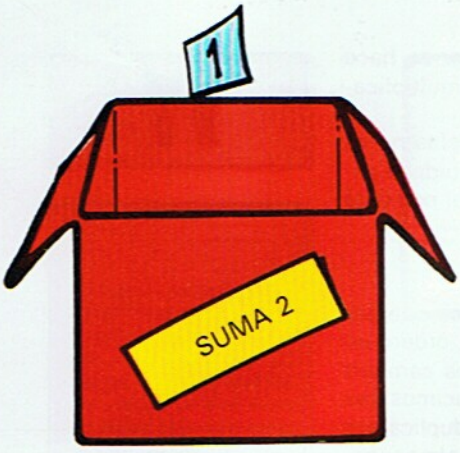


¡!

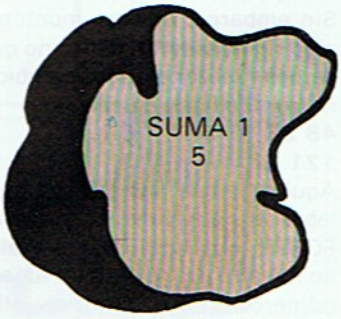
Cuando realizamos una operación (aritmética, visualización o de condición) con los datos de la pila, los que son utilizados se pierden y son sustituidos por el resultado, en el caso de la operación, o por un cero si la condición es falsa o un uno si es verdadera.

*

En la pila es donde el FORTH guarda los datos de donde los extraerá cuando sea necesario operar.



VARIABLE



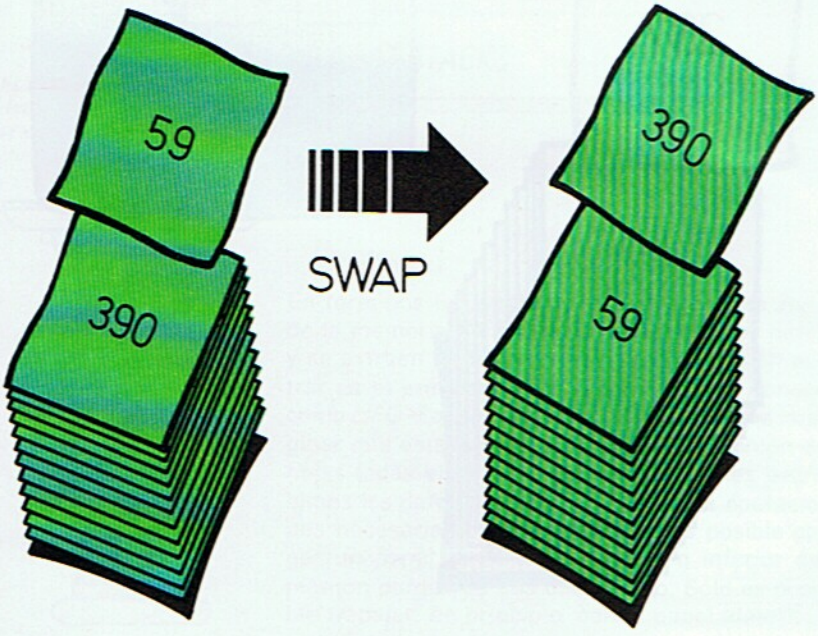
CONSTANTE

El FORTH, al igual que en otros lenguajes, nos ofrece la posibilidad de guardar los datos en variables o en constantes.

Después de introducir los datos en la pila, **OVER** nos duplica el penúltimo dato (el 43) con lo que la pila quedará **35 43 12 43**. Sin embargo, **DROP** elimina el último de la pila, por lo tanto tendremos otra vez la situación inicial **35 43 12**, como se puede comprobar al imprimirlos.
Podemos incluso cambiar la colocación de los números de la pila, pero claro, de los de arriba, para evitar que se nos caiga nuestro montoncillo de papeles. Para ello tenemos **SWAP** que intercambia las dos hojas de arriba y **ROT** que se atreve a cambiar de lugar los tres papeles superiores, de tal manera que el que se encuentra en tercer lugar (contando por arriba) le extrae y coloca encima de todos, por tanto el que estaba en segundo pasará a ser el tercero y el primero el segundo.

67 54 32 ROT CR...
54 32 67 OK
32 51 28 SWAP CR...
32 28 51 OK

Una de las formas de mover los datos de la pila es con **SWAP**, *que nos intercambia el orden de los dos últimos datos de la pila.*



CONSTANTES Y VARIABLES

FORTH también conoce otras formas de guardar valores para luego trabajara con ellos, y que son comunes a otros lenguajes informáticos; son las constantes y las variables, pero es necesario decir que para operar con ellas siempre deben estar en nuestra pila de trabajo (conocida mejor por pila o *stack* de usuario).

5 CONSTANT SUMA1
1 VARIABLE SUMA2
SUMA 2 §
SUMA 1 +

Veamos en que consiste este ejemplo: en primer lugar definimos una constante y le damos el valor **5** que se almacena en **SUMA1.** Posteriormente hemos hecho lo mismo con una variable y su valor, **1**, se guarda en **SUMA2.** Lo siguiente es guardar el valor de la variable en la pila del usuario, algo necesario si queremos operar con ella, para lo cual nos valemos de la palabra **§**. Por último, entramos el valor de la constante en la pila y realizamos la suma de ambos.
Ahora tenemos dos opciones: o visualizar el resultado, con lo cual ya sabemos que lo perdemos de la pila, o volverlo a guardar en la variable para cuando nos sea necesario usar de nuevo su valor. Como lo primero ya sabemos hacerlo vamos a efectuar la segunda opción. Teclee ahora:

SUMA2 !

Ya tenemos guardado el valor superior de la pila en la variable **SUMA2.** ¿Qué no nos lo creemos? Pues a teclear:

SUMA2 §.

O bien:

SUMA2 ?

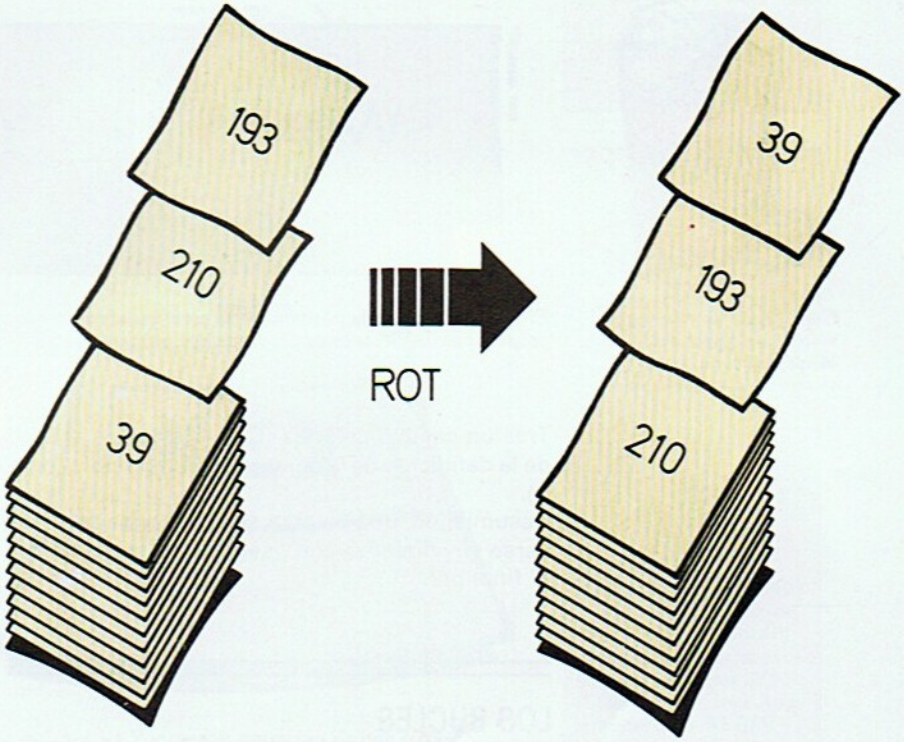
En ambos casos aparecerá «6», que es el valor esperado. Ambas expresiones son equivalentes, con lo cual la segunda se explica por la primera.

DEFINIMOS NUESTRAS PALABRAS

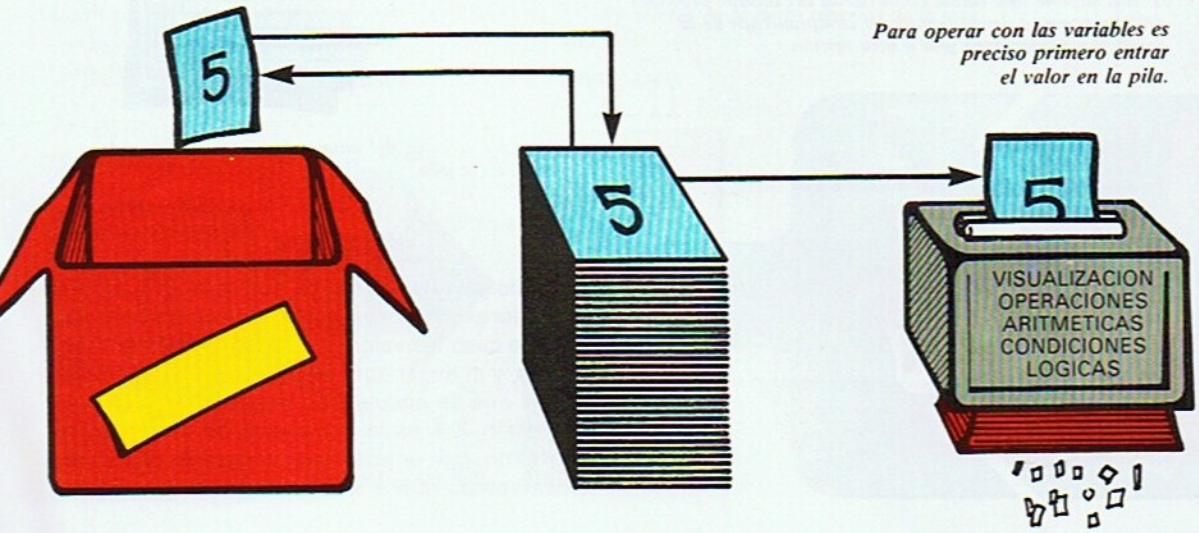
Ya comentamos al principio del artículo que el FORTH trabaja con palabras, de tal manera, que no se considera que existan programas en FORTH sino palabras. Además podemos crear nosotros nuestras propias palabras para realizar todas aquellas operaciones necesarias.
Podremos definir palabras nuevas que podrán estar formadas por otras (ya sean también creadas por nosotros o de las que FORTH incorpora). Una vez definidas, con llamarles por su nombre basta para que se ejecuten y aparezcan los resultados esperados.
Pasemos a la práctica y definamos nuestra primera palabra:

```
:LETRA
CR KEY
DUP EMIT CR
3 + EMIT CR ;
```

Si suena a chino no nos preocupemos que ahora intentaremos que todo quede claro. En primer lugar vemos : cuya función es indicar que vamos a definir una palabra nueva a la cual llamaremos por el nombre indicado posteriormente **(LETRA)**. La función de la palabra **KEY** es esperar la pulsación de una tecla y guardar su código ASCII en la pila del usuario, trabajando con él como si fuera un número. Así, si hemos pulsado la **A** en la pila se guardará el número **65**. Tras introducir la tecla, la palabra continúa ejecutándose. Posteriormente, duplicamos el valor del último número de la pila, que será el código ASCII de la tecla pulsada. A continuación imprimimos la letra cuyo código se encuentre en la parte superior de la pila. Esto lo hacemos con ayuda de **EMIT**. Así, continuando con nuestro ejemplo, se imprimirá **A** porque en la pila tenemos su código.
El siguiente paso está claro, puesto que sumamos **3** a la pila, con lo cual el valor de ella pasa a ser ahora de **68**. Después le indicamos que imprima la letra cuyo código se encuentra en la parte superior de la pila, que, en este ejemplo es **D**.



ROT *también nos ofrece la posibilidad de intercambiar los valores de la pila. En este caso se intercambian los tres últimos datos del stack.*



Para operar con las variables es preciso primero entrar el valor en la pila.

¡!

Los gráficos y el sonido son iguales que en el BASIC, salvo que los datos se anteponen a la palabra separados, por un espacio.

*****

Para imprimir una cadena alfanumérica se escribe primero un punto (.), y, sin separación, la cadena entrecomillada cuyo primer carácter debe ser siempre un espacio.

Con **DROP** *se elimina el valor del último dato introducido sin ser visualizado ni operado.*

El FORTH nos da la posibilidad de crear palabras propias para realizar la función que nos interesa.

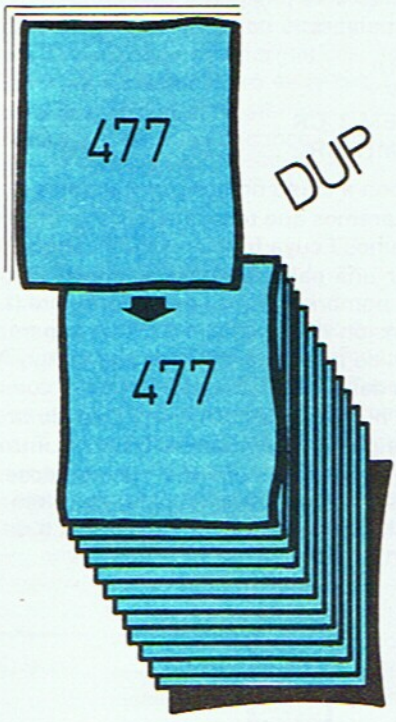
Tras un cambio de línea (**CR**), indicamos el final de la definición de la palabra con un punto y coma (;).
Resumiendo, una palabra se caracteriza por indicarse su comienzo con : seguido de su nombre y el final por ;.

LOS BUCLES

Dentro de este grupo general de órdenes, cuya característica común es provocar la repetición de un grupo de comandos, podemos distinguir los no condicionales y los condicionales (que a diferencia del grupo anterior, una condición limita la continuidad del bucle).
Dentro del primer grupo tenemos las palabras **DO...LOOP**, que se presentan siempre asociadas y entre ellas se introducen las instrucciones a repetir. Para una mejor comprensión diremos que son totalmente equivalentes a los bloques **"FOR/TO...NEXT"** del BASIC. Veamos el siguiente ejemplo:

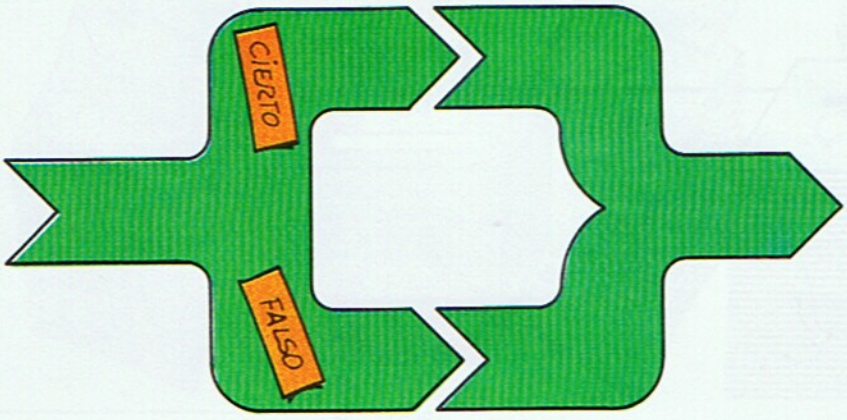
: PRESENTACION 6 0 DO ."SOY FORTH" CR 2 +LOOP ;

Al ejecutar esta palabra (llamada «PRESENTACION») podemos comprobar que se imprime tres veces el mensaje «SOY FORTH» debido al bucle utilizado. Los dos primeros números, **6 0**, son los limitadores en la repetición de las órdenes, así en este caso los valores variarán desde cero hasta seis, y al llegar aquí se saltará al bloque de palabras que se encuentran después de **LOOP**. La expresión **2 +** es la encargada de indicar el incremento, que en este caso lo hará de dos en dos (observemos que + se encuentra unido directa-



DUP *repite el valor del último dato introducido y lo deposita en la pila.*

IF nos supone una salida condicional del bloque principal del programa, y dependiendo del cumplimiento de la condición se ejecutará una u otra opción.



mente a **LOOP**). Se hubiera podido simplificar este ejemplo con haber hecho **3 0 DO** ... y haber suprimido el incremento.
Conviene fijarse en la forma que se debe emplear para imprimir cadenas alfanuméricas, sobre todo porque ninguna de las dos guías FORTH de los programas mencionados menciona como hacerlo. Después de **.** y **sin separación** se coloca la cadena entrecomillada y cuyo primer carácter debe ser siempre un espacio.
El FORTH dispone también de la palabra condicional **IF** que, salvo su sintaxis, es en todo equivalente al BASIC.

```
: IGUALDAD CR KEY CR KEY > IF
." ES MAYOR EL PRIMERO" ELSE
." ES MAYOR EL SEGUNDO" ENDIF ;
```

Los términos a comparar se deben colocar en la pila y justo antes de la palabra **IF** (aquí se comparan los dos datos introducidos por teclado) se indica el signo de la condición (<>=). Si se cumple, se ejecuta lo que está a continuación y en el caso contrario salta y es ejecutado lo que se encuentra después del **ELSE**. Por último, se indica el final del bloque **IF** con **ENDIF**.
Dentro de los bucles condicionales, propiamente dichos, tenemos dos opciones: **BEGIN ... UNTIL** y **BEGIN ... WHILE ... REPEAT**. La diferencia entre ambas se encuentra en que mientras en la primera el bucle se está ejecutando hasta que se cumpla la condición (es decir, que el número superior del *stack* sea un cero), en el segundo se ejecuta continuamente hasta que la condición deje de cumplirse. Veamos sus sintaxis:

```
: LETRA CR BEGIN KEY 65
— UNTIL ." NO HAS APRETADO LA A" ;
```

BEGIN indica el comienzo del bucle en donde se introduce una tecla y a cuyo código se le resta 65. Si se ha pulsado **A** el valor que queda en la parte superior de la pila es **0**, por tanto vuelve a ejecutarse el bucle. Cuando no se apriete la tecla salta y se imprime el mensaje del final.

La palabra KEY *espera la pulsación de una tecla e introduce su código ASCII en la parte superior de la pila.*

```
: OPERACIONES CR BEGIN 10 20
KEY 65 = WHILE DROP + ." 10+20=" .
REPEAT DROP * ." 10*20=" . ;
```

En este caso, dentro del bucle introducimos dos números, el código de una letra y un cuarto número. Cuando se cumpla la condición (pulsemos **A**) se imprimirá la suma de los dos números y se repetirá de nuevo el bucle. Si no se cumple se ejecuta lo que se encuentra después del **REPEAT** abandonando el bucle.

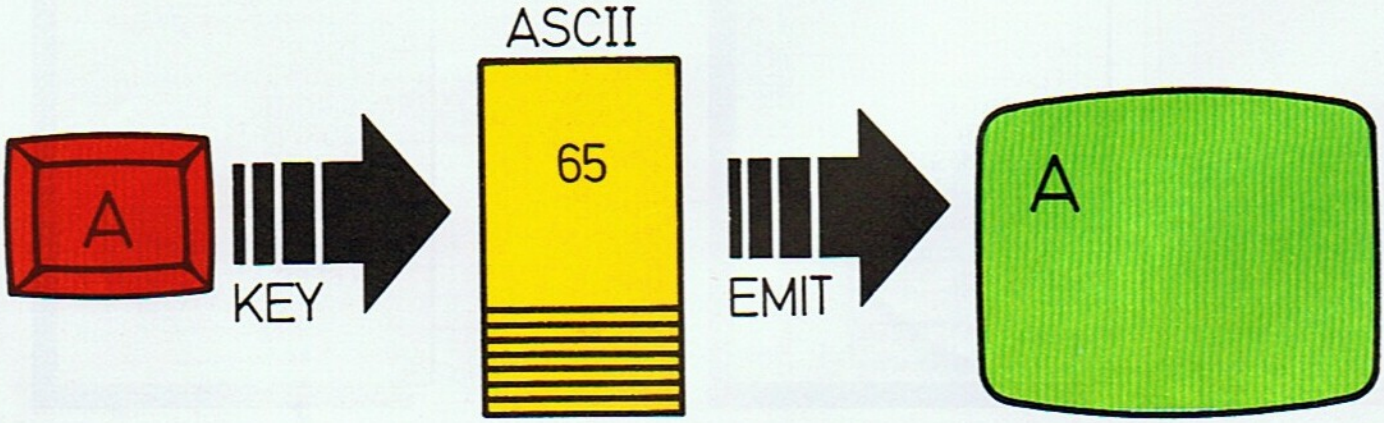
EMIT *es la contraria a* KEY: *imprime la letra cuyo código ASCII está indicado por el número que se encuentra en la parte superior de la pila.*

¡!

Una palabra de FORTH se indica porque en su comienzo se escribe : y posteriormente su nombre. Se termina la definición con ;.

*

Los bucles son de dos tipos: los condicionales y los no condicionales. Los primeros tienen la forma **DO ... LOOP** y los segundos pueden ser **BEGIN ... UNTIL** o **BEGIN ... WHILE ... REPEAT**.



¡!

Las peculiaridades o los inconvenientes más característicos de este lenguaje es que no «sabe» operar con decimales ni conoce otras operaciones que las cuatro básicas (+, –, *, /).

*

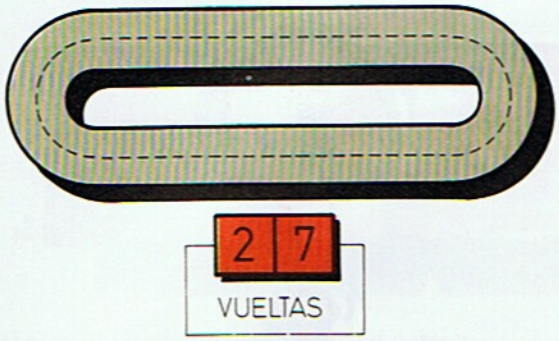
En **BEGIN ... UNTIL** el bucle se está ejecutando hasta que se cumpla la condición. En **BEGIN ... WHILE ... REPEAT** el bucle se ejecuta hasta que la condición deje de cumplirse.

*

Las funciones **KEY** o **EMIT** son contrarias; mientras que la primera espera a que sea pulsada una tecla y guarda el valor de su código ASCII en la pila, la segunda coge el dato superior del *stack*, lo toma como código ASCII e imprime el carácter correspondiente.

*

Tenemos la posibilidad de crear nuestras propias palabras formadas por otras creadas por nosotros o de la que el lenguaje incorpora.



La única limitación de los bucles no condicionales es el número de veces que se debe repetir las instrucciones.

LOS GRAFICOS Y LOS SONIDOS

Sería ilógico pensar que un lenguaje con la finalidad del FORTH no tuvira posibilidad gráfica y sonora. Pero si le ha parecido difícil lo visto hasta ahora, este es el momento de descansar dado que, en este aspecto, los comandos coinciden con los del BASIC SPECTRUM y la diferencia se encuentra que hay que anteponer los valores separados por un espacio. Lo único reseñable es que en la versión FORTH de *ABERSOFT* el comando **DRAW** no se refiere a los incrementos de X e Y (como es en el BASIC SPECTRUM) sino a la posición absoluta donde terminará el trazo.

Los comandos gráficos y sonoros son iguales a los del BASIC, sólo cambia la posición de los datos.

BASIC

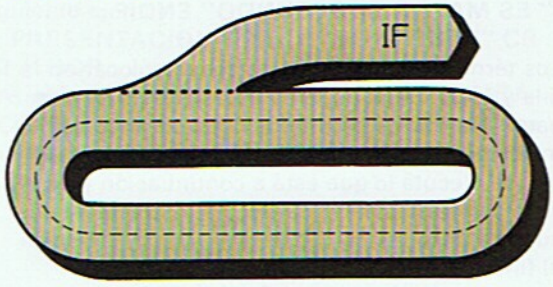
```
INK a
PAPER a
OVER a
FLASH a
BRIGHT a
BORDER a
INVERSE a
PLOT a, b
DRAW a, b
CIRCLE a, b
BEEP a,b
```

FORTH

```
a INK
a PAPER
a OVER
a FLASH
a BRIGHT
a BORDER
a INVERSE
ab PLOT
ab DRAW
ab CIRCLE
ab BEEP
```

YA TERMINAMOS

¿Qué le ha parecido esta introducción al FORTH-SPECTRUM?: cualquier cosa menos un camino de rosas. Este lenguaje a todas luces no es para los que se inicien en la informática, aunque sus autores propagaran que también servía para eso. Es indudable que introducir la idea de *stack* y de su manejo a un novicio se puede hacer bastante cuesta arriba.



En un bucle condicional hay una condición lógica que cuando se cumple se sale de él.

Con respecto a las dos versiones FORTH para Spectrum queda por decir que son bastante similares entre sí y, además de las ya comentadas, aportan de diferencias que la de INVESTRONICA tiene su guía en castellano y el Editor (parte del FORTH que sirve para facilitar la corrección y realización de las palabras) hay que cargarlo aparte del intérprete, mientras que en el de *ABERSOFT*, la guía es en inglés y el Editor se carga al mismo tiempo que el FORTH.

MOVIENDO 16 BITS

EN el capítulo anterior discutíamos en detalle el grupo de carga de 8 bits. Nuestro próximo objetivo consistirá en efectuar otro tanto, con las instrucciones encargadas de gestionar la transferencia de 16 bits entre registros dobles y la memoria de nuestro ordenador.
A ellas uniremos un grupo muy especial, el cual actúa sobre una importante zona de la RAM del Spectrum denominada *stack* o pila, encargada de conservar ciertos datos de interés durante la ejecución de un programa, para su posterior utilización.

PAR	dd	qq
BC	00	00
DE	01	01
HL	10	10
SP	11	—
AF	—	11

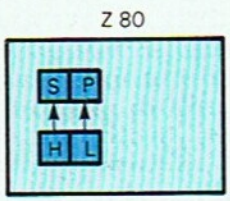
Decodificación seguida por el Z80 para reconocer los pares de registros indicados.

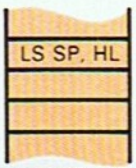
CARGA DE 16 BITS

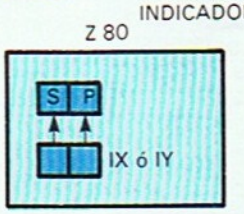
Si observamos la tabla que lleva este nombre al comienzo de nuestros capítulos dedicados al código máquina, la encontramos mucho menos poblada de lo que estaba la de carga de 8 bits, sobre todo en lo referente a transferencia de datos entre registros dobles. Solamente cuando el destinatario de la información es el *stack pointer* SP es posible seleccionar algunos pares de registros como origen de ésta.
La última fila y la última columna están señaladas con PUSH y POP, respectivamente, instrucciones que discutiremos en breve. Pero centrémonos en el resto de la tabla. Esta zona constituye el grupo de carga propiamente dicho y, por supuesto, el código mnemónico asociado es LD. El formato es idéntico al grupo anterior:

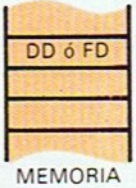
ENSAMBLADOR	CODIGO MAQUINA	N.º BYTES	CICLOS RELOJ	CICLOS MAQUINA
LD SP, HL	1 1 1 1 1 0 0 1	1	6	1
LS SP, IX	1 1 0 1 1 1 0 1 1 1 1 1 1 0 0 1	2	10	2
LS SP, IY	1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1	2	10	2

INDICADORES NO AFECTADOS







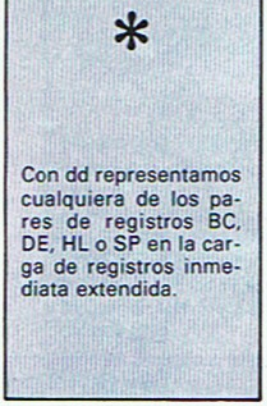


LS SP, IX
LS SP, IY

En la tabla de carga de 16 BITS la última fila y última columna están señaladas con PUSH y POP.

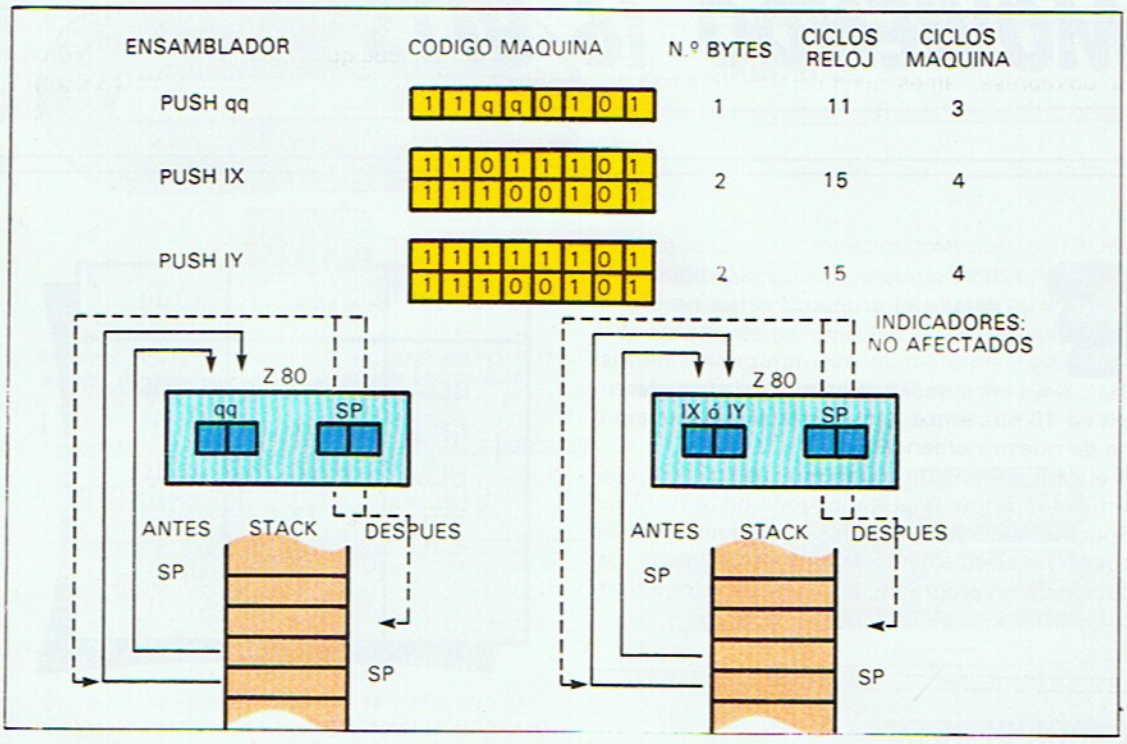
Las instrucciones PUSH y POP están ligadas con una zona de la memoria RAM de nuestro Spectrum denominada *stack* o pila.

Al grupo de registros dd es posible añadir los índices IX e IY, pues es factible almacenar directamente en ellos un entero nn.



Con dd representamos cualquiera de los pares de registros BC, DE, HL o SP en la carga de registros inmediata extendida.

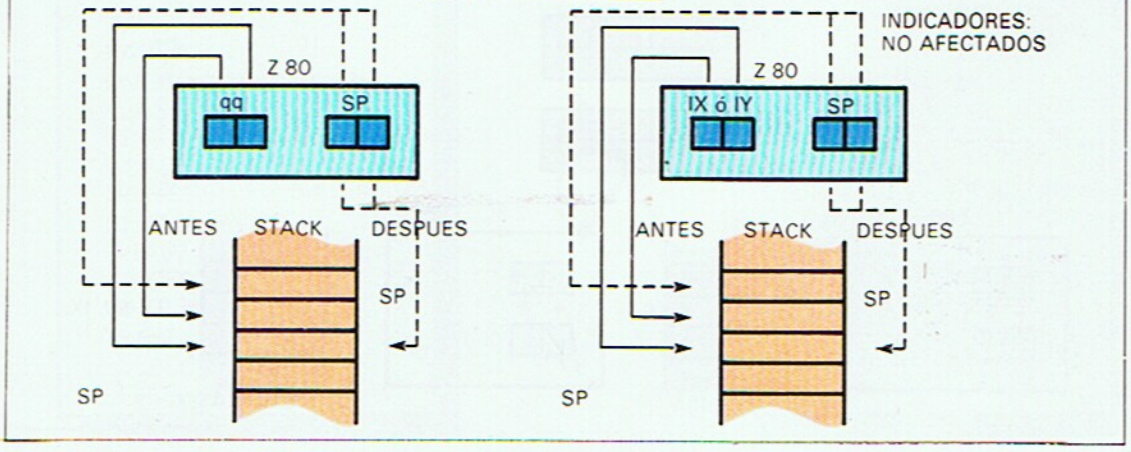
ENSAMBLADOR	CODIGO MAQUINA	N.º BYTES	CICLOS RELOJ	CICLOS MAQUINA
PUSH qq	1 1 q q 0 1 0 1	1	11	3
PUSH IX	1 1 0 1 1 1 0 1 1 1 1 0 0 1 0 1	2	15	4
PUSH IY	1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1	2	15	4



LD DESTINO,ORIGEN o LD DESTINO,DATO

pero con la diferencia, ya advertida, de que tanto ORIGEN, DESTINO, así como DATO son registros, posiciones de memoria o cantidades de 16 bits. Lo que es lo mismo: abarcan dos octetos.
Para encontrar el código hexadecimal correspondiente a una instrucción entraremos en la tabla por la fila DESTINO y buscaremos su intersección con la columna ORIGEN. De nuevo, si en este lugar se haya un cuadro en blanco, la instrucción planteada no está contemplada en el juego del Z80. Efectuemos la siguiente clasificación:

ENSAMBLADOR	CODIGO MAQUINA	N.º BYTES	CICLOS RELOJ	CICLOS MAQUINA
POP qq	1 1 q q 0 0 0 1	1	10	3
POP IX	1 1 0 1 1 1 0 1 1 1 1 0 0 0 0 1	2	14	4
POP IY	1 1 1 1 1 1 0 1 1 1 1 0 0 0 0 1	2	14	4



1. Carga en registros.

1.1. Inmediata extendida: **LD dd,nn**

Con dd representamos cualquiera de los pares de registros BC, DE, HL o SP. En la figura se muestra la decodificación seguida por la CPU de nuestro Spectrum para reconocerlos. Tras ejecutarla, el valor entero determinado por los dos bytes nn, es decir una cantidad comprendida entre 0 y 65535, se «copia» en el par de registros dd. Has de tener presente el orden de los octetos nn: primero el de menor peso y en segundo lugar el más significativo.

A los registros dd es posible añadir los índice IX e IY, pues es factible almacenar directamente en ellos un entero nn mediante las instrucciones **LD IX,nn** y **LD IY,nn.**

Como en el capítulo anterior, en las tablas correspondientes está recogida toda la información accesoria sobre cada grupo particular de instrucciones.

1.2. Carga extendida LD dd,(nn)

Para analizar el comportamiento de esta instrucción, lo mejor es recurrir a un ejemplo en el cual consideraremos que el par de registros dd es el BC. Supongamos, por tanto, que deseamos procesar **LD BC,(5FA6h)** o en su correspondiente expresión decimal **LD BC,(24486d).** Efectuemos también el supuesto de que la posición de memoria 24486 almacena el valor 176 y la 24487, conserva un 209.

Pues bien, tras leer el código de operación de dicha instrucción, el microprocesador procederá de la manera siguiente: buscará la posición de memoria 24486 y trasladará el dato allí almacenado (176), al octeto menos significativo del par de registros destino.

A continuación, volverá a acceder a la memoria para recoger el dato conservado en la dirección siguiente, es decir, en la 24487 y lo copiará en el byte de mayor peso del par BC, el cual, tomará el valor 176+209*256=53680.

Puedes comprobarlo tecleando y ejecutando el siguiente programa, el cual introduce en las direcciones del ejemplo los valores señalados y a continuación almacena en la memoria la rutina en C/M:

Dirección	Ensamblador
24488	LD BC,(24486d)
24492	RET

```
10 FOR I=24486 TO 24492
20 READ A: POKE I,A:NEXT I
30 PAUSE 0: PRINT USR 24488
40 DATA 176,209,237,75,166,95,201
```

Las instrucciones de este tipo ocupan cuatro bytes, pero dentro del juego del Z80, sus fabricantes decidieron dar un trato preferencial a la pareja HL, de modo tal, que cuando dd coincide con el par anterior (LD HL,(nn)) puede implementarse en sólo tres bytes. Por tanto, es ejecutable en dos formatos distintos, pero en este último se ahorra memoria y tiempo.

También es factible que el destinatario de la información sea cualquiera de los registros índice mediante el uso de las instrucciones **LD IX,(nn)** o **LD IY,(nn)** cuyo funcionamiento es idéntico al de las consideradas en este subgrupo.

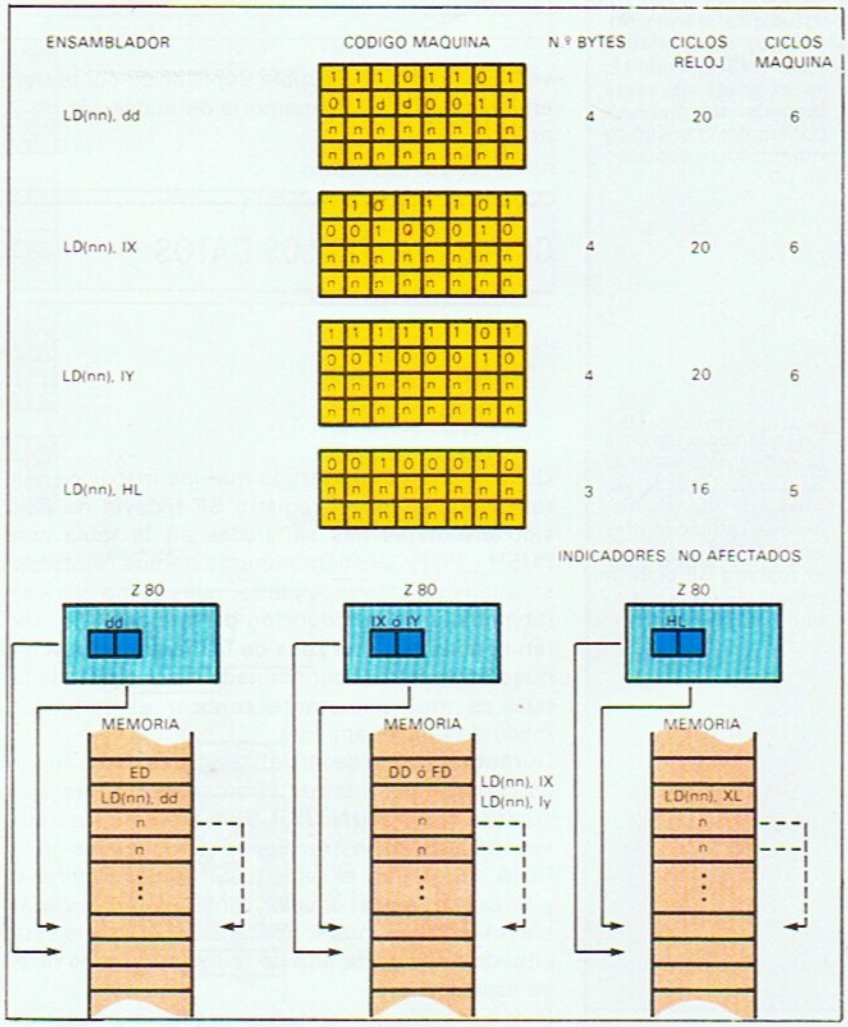
2. Carga en memoria.

2.1. Carga extendida: **LD (nn),dd**

Este grupo puede considerarse inverso del 2.2, el cual manejaba direccionamiento extendido para copiar la información almacenada en una pareja de registros en la memoria de nuestro ordenador. Ahora el proceso es el contrario, es decir, la CPU toma el byte de menor peso contenido en el par dd y lo duplica en la celda de memoria direccionada por nn. A continuación, lee el octeto más significativo conservado en dd y lo copia en la siguiente posición a nn.

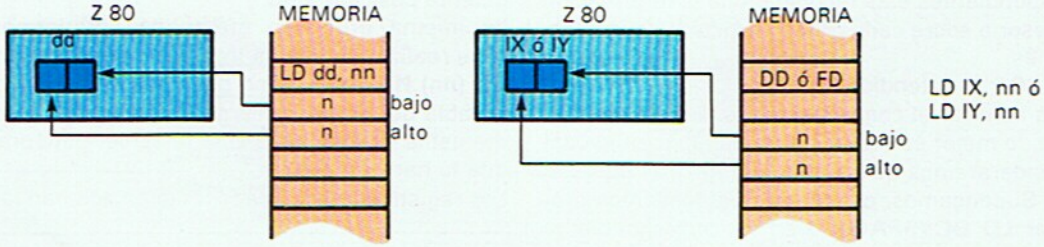
La misma precisión efectuada anteriormente cabe realizar cuando la instrucción a procesar es **LD (nn),HL,** pues como podemos comprobar en la tabla del grupo de carga de 16 bits es implementable en tres bytes, aparte del formato previo que lo hace en cuatro.

Los registros índice IX e IY también aceptan la in-



ENSAMBLADOR	CODIGO MAQUINA	N.º BYTES	CICLOS RELOJ	CICLOS MAQUINA
LD dd, nn	0 0 0 0 0 0 0 1 n n n n n n n n n n n n n n n n	3	10	3
LD IX, nn	1 1 0 1 1 1 0 1 F0 0 1 0 0 0 0 1 n n n n n n n n n n n n n n n n	4	14	4
LD IY, nn	1 1 1 1 1 1 0 1 0 0 1 0 0 0 0 1 n n n n n n n n n n n n n n n n	4	14	4

INDICADORES: NO AFECTADOS



La tabla de carga de 16 BITS, menos las columnas señaladas PUSH y POP, constituye el grupo de carga propiamente dicho, y por supuesto el código mnemónico asociado es LD.

Cuando conectamos a la red el ordenador el Sistema Operativo, mediante las instrucciones almacenadas en la **ROM**, carga en el registro SP el valor contenido en la variable RAMTOP-1.

versa, es decir, es factible duplicar su contenido en las posiciones de memoria definidas por nn y nn+1.

CONSERVANDO LOS DATOS

Quizá habremos advertido que las instrucciones relacionadas con el registro SP todavía no han sido discutidas (las señaladas en la tabla con PUSH y POP). Deliberadamente hemos retrasado su análisis hasta este punto, pues como comentábamos en la introducción de este capítulo, están ligadas con una zona de la memoria RAM de nuestro Spectrum denominada *stack* o pila, de la cual es muy importante conocer el particular modo de funcionamiento.

Durante la rutina de inicialización, es decir, cuando conectamos a la red el ordenador o tras ejecutar la orden **RUN USR 0,** el Sistema Operativo mediante las instrucciones almacenadas en la ROM, cargan en el registro SP *(Stack Pointer* o puntero de la pila) el valor contenido en la variable RAMTOP menos 1. Por tanto, al principio está situado en la parte alta de la memoria y no ocupa espacio.

Pero a medida que el microprocesador o nosotros mismos vamos introduciendo datos en él, «crece» desde allí hacia las direcciones de memoria menores, es decir, el valor del registro doble SP disminuye a medida que el *stack* aumenta de tamaño, y viceversa.

Esta circunstancia debe ser tenida muy en cuenta en la construcción de programas, pues un aumento incontrolado del *stack* puede traer como consecuencia la invasión por su parte del área de BASIC u otras zonas que en principio no hemos previsto para él, quedando adulterados por este motivo los resultados previstos.

Sobre la pila es posible efectuar dos operaciones: INTRODUCIR bytes (PUSH) o EXTRAERLOS (POP). Para ejecutarlas es necesario especificar siempre un par de registros como origen o destino de los datos intercambiados. Por tanto (es imprescindible tenerlo en cuenta), la transferencia de información entre registros y la pila o en sentido contrario es siempre de DOS en DOS bytes.

En la figura podemos encontrar los formatos bajo los cuales es posible implementar las instrucciones PUSH y POP. Veamos las operaciones llevadas a cabo por el Z80 al ejecutarlas. Supongamos que deseamos introducir (PUSH) un par de datos en la pila:

1) Se disminuye en 1 el contenido del par SP.
2) Se copia el contenido del byte de orden alto del par de registros origen en la posición de memoria indicada por SP.
3) Se decrementa SP.
4) Finalmente, el valor almacenado en el octeto de menor peso del par origen se duplica en la dirección apuntada por SP.

Ahora supongamos que queremos realizar el proceso inverso, es decir, extraer (POP) información de la pila para ser almacenada en un par de registros:

1) El contenido de la posición de memoria direccionada por SP se carga en el octeto de orden bajo del par de registros elegido.
2) Se le añade 1 al contenido del SP.
3) El byte indicado ahora por SP se transfiere al octeto más significativo del par correspondiente.
4) Nuevamente es incrementado en 1 el contenido del registro SP.

En la jerga informática esta forma de tratar los datos se denomina LIFO *(Last In, First Out:* último en entrar, primero en salir) y su utilidad es obvia si consideramos una cierta cantidad de información, la cual desborde la capacidad de los registros, y además, la deseemos mantener estructurada de manera tal que tengamos fácil y rápido acceso al recuperarla.

Algunos ordenadores disponen de la denominada pila del usuario para todas estas labores. Desgraciadamente el Spectrum sólo cuenta con la pila de máquina, y sobre ella actuamos nosotros con las instrucciones PUSH y POP.

Pero el microprocesador también necesita almacenar ciertos valores de interés para él, por ejemplo, la dirección a donde regresar cuando termine de ejecutar una subrutina. Por ello, cuando hagamos uso de la pila debemos asegurarnos que el número de instrucciones PUSH y POP ejecutadas es el mismo.

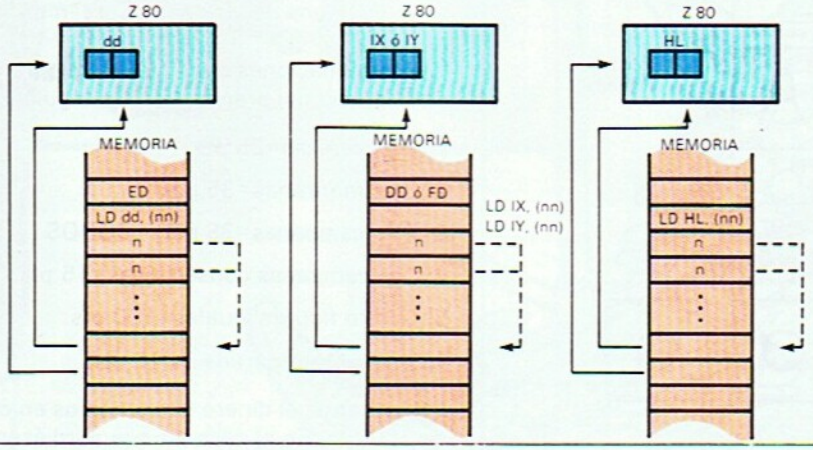
En otro caso el Z80 tomará los dos bytes indicados por el registro SP como dirección de retorno y el resultado puede convertirse en imprevisible.

Por otra parte, de la misma manera que en BASIC al ejecutar el comando **CLEAR** se modifica el RAMTOP del Sistema, en C/M es posible desplazar el puntero del stack SP a la posición de memoria determinada por un par de registros. Para ello basta ejecutar cualquiera de las instrucciones siguientes: **LD SP,HL, LD SP,IX** ó **LD SP,IY.**

La primera, como podemos comprobar en la tabla correspondiente es una instrucción rápida que tan solo ocupa un byte de memoria. Tras ejecutarla, el SP señalará a la posición de memoria definida por el contenido del par HL. Con esto queda cerrada la discusión emprendida en el capítulo anterior sobre las instrucciones de carga.

ENSAMBLADOR	CODIGO MAQUINA	Nº BYTES	CICLOS RELOJ	CICLOS MAQUINA
LD dd, (nn)	1 1 1 0 1 1 0 1 0 1 d d 1 0 1 1 n n n n n n n n n n n n n n n n	4	20	6
LD IX, (nn)	1 1 0 1 1 1 0 1 0 0 1 0 1 0 1 0 n n n n n n n n n n n n n n n n	4	20	6
LD IY, (nn)	1 1 1 1 1 1 0 1 0 0 1 0 1 0 1 0 n n n n n n n n n n n n n n n n	4	20	6
LD HL, (nn)	0 0 1 0 1 0 1 0 n n n n n n n n n n n n n n n n	3	15	5

INDICADORES NO AFECTADOS



¡!

En la carga en extendida en memoria, la CPU toma el byte de menor peso, contenido en el par dd y lo duplica en la celda de memoria direccionada por nn.

*

Los registros índice IX e IY aceptan la inversa, es decir, es factible duplicar su contenido en las posiciones de memoria definidas por nn y nn+1.

TRAGAPERRAS

ESTE programa es la solución al problema que padecemos la casi totalidad de la humanidad: no nos gusta perder dinero en el juego. Con TRAGAPERRAS, podremos arriesgar nuestros ahorros, sin perder un solo duro utilizando el dinero imaginario que el Spectrum pone a nuestra disposición.

Para que cada una de las figuras comience a girar es necesario pulsar ENTER *cuando lo indique el ordenador.*

EL PROGRAMA

TRAGAPERRAS es un sencillo programa en el cual predomina la suerte y la aleatoriedad. Su manejo es muy sencillo; basta con seguir al pie de la letra las instrucciones que nos surjan en cada momento en la pantalla.

¿El objetivo del juego? No puede ser más simple; simula justamente lo que su propio nombre indica: una máquina tragaperras de casino.

El programa nos muestra cuatro ventanas en las cuales irán apareciendo en secuencia aleatoria, y en un espacio de tiempo determinado, una serie de figuras; dependiendo del orden final de las cuales obtendremos más o menos ingresos o... nada si la suerte no nos acompaña.

Para que cada una de las figuras comience a girar es necesario pulsar **ENTER** cuando así lo indique el ordenador. Una vez pulsada dicha tecla, el programa principal pone en funcionamiento una rutina que durante un intervalo de tiempo predeterminado irá seleccionando aleatoriamente en cada una de las cuatro ventanitas una figura.

Las combinaciones a las que el Spectrum concede la gracia del premio, son las siguientes:

1: Tres cerezas=35 pts.

2: Tres manzanas=35 pts.

3: Tres campanas=35 pts. + BONOS

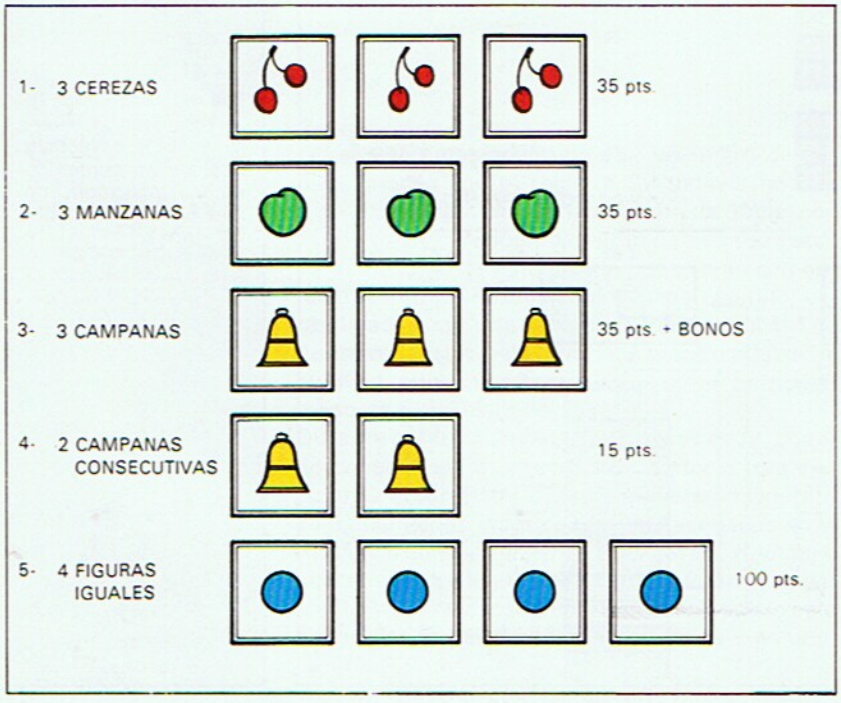
4: Dos campanas consecutivas=15 pts.

5: Cuatro figuras iguales=100 pts.

* El precio por partida es 15 pts.

Inicialmente, el dinero que tenemos en cuenta es de 100 pts. En el caso en que perdiésemos todo

Las combinaciones a las cuales el Spectrum concede la gracia de premio son las representadas en el dibujo.



el dinero acumulado, podremos volver a probar suerte con solo pulsar una tecla. Si por el contrario la suerte nos favorece, podremos seguir jugando siempre y cuando no sobrecalentemos los circuitos de la máquina tragaperras. Esto nos puede sonar a chiste, pero a ver quien es el guapo que se atreve a pasar el tope de 9.500 pts. que el Spectrum se ha autoimpuesto.

```
 10 REM ***********************
 20 REM * J.M.MAYORAL SERRANO *
 30 REM ***********************
 40 REM * TRAGAPERRAS    1985 *
 50 REM ***********************
 60 REM [INV.] INIC VARIABLES [TRUE]
 70 RANDOMIZE
 80 LET SW=0
 90 PAPER 0: BORDER 0: INK 9
100 CLS
110 LET DIN=100
120 DIM A$(3,4): DIM B$(3,4): DIM A(4): DIM Q(4)
130 IF SW THEN  GO TO 200
140 PRINT AT 10,5;"UN MOMENTO POR FAVOR"
150 GO SUB 1250
160 CLS
170 FOR N=1 TO 4
180 PRINT  PAPER 2; INK 6;AT 4,N*3;N
190 NEXT N
200 REM [INV.] DIBUJO MAQUINA [TRUE]
210 FOR M=22 TO 98 STEP 24
220 PLOT M,129: DRAW 19,0
```

```
 230 DRAW 0,-19
 240 DRAW -19,0
 250 DRAW 0,19
 260 NEXT M
 270 PLOT 18,131
 280 DRAW 99,0
 290 DRAW 0,-23
 300 DRAW -99,0
 310 DRAW 0,23
 320 PLOT 14,98
 330 DRAW 107,0
 340 DRAW 0,-13
 350 DRAW -107,0
 360 DRAW 0,13
 370 PLOT 10,50
 380 DRAW 0,100
 390 DRAW 10,6
 400 DRAW 95,0
 410 DRAW 10,-6
 420 DRAW 0,-100
 430 DRAW -115,0
 440 PRINT AT 12,4;'FRUIT'
 450 PRINT AT 14,6;'MACHINE'
 460 PRINT AT 10,2; INK 6;'8888888888888'
 470 GO SUB 720
 480 IF RND>.87 AND PR<100 THEN  GO SUB 500
 490 GO TO 470
 500 REM [INV.] AVANCES [TRUE]
 510 FOR T=1 TO 7: BEEP .1,50-T*7: NEXT T
 520 PRINT AT 8,19; FLASH 1; INK 9;'AVANCES'
 530 LET CONTADOR=0
 540 PRINT AT 21,2;'PULSA EL NUMERO DE CASILLA'
 550 IF CONTADOR=4 THEN  PRINT AT 21,0;'
                    ': GO TO 640
 560 INPUT Q$
 570 IF Q$='' THEN  LET CONTADOR=4: GO TO 550
 580 IF CODE Q$<49 OR CODE Q$>52 THEN  BEEP 1,-10: GO
 TO 560
 590 LET CONTADOR=CONTADOR+1
 600 LET Q=VAL Q$(1)
 610 LET Q(Q)=Q
 620 PRINT AT 9+Q,18; PAPER RND*4; INK 9; FLASH 1;'PA
RADO >';Q;' '
 630 GO TO 560
 640 PRINT AT 8,18;'                ';AT 10,18;'
       ';AT 11,18;'                ';AT 12,18;'
      ';AT 13,18;'
 650 RETURN
 660 REM [INV.] COG. NUM. [TRUE]
 670 PRINT AT 6,3;A$(A(1));AT 7,3;B$(A(1))
 680 PRINT AT 6,6;A$(A(2));AT 7,6;B$(A(2))
 690 PRINT AT 6,9;A$(A(3));AT 7,9;B$(A(3))
 700 PRINT AT 6,12;A$(A(4));AT 7,12;B$(A(4))
 710 RETURN
 720 REM [INV.] NUEVO JUG. [TRUE]
 730 LET DIN=DIN-15
 740 LET PR=0
 750 LET O$='': LET P$=''
 760 PRINT AT 4,18; FLASH 1; PAPER 1; INK 7;'PULSA EN
TER'
 770 IF INKEY$<>'' THEN  GO TO 770
 780 IF INKEY$<>CHR$ 13 THEN  GO TO 780
 790 PRINT AT 4,18;'                ';AT 5,19;'
    ';AT 6,21;'
 800 PRINT AT 19,0;'
 810 PRINT AT 20,0;'
 820 PRINT AT 10,2; PAPER 6;'
 830 PRINT AT 2,18; PAPER 1; INK 9;'DINERO =  ';DIN;
PAPER 0;' '
 840 FOR Z=1 TO 8
 850 FOR B=1 TO 4
 860 IF Q(B)=B THEN  GO TO 890
 870 BEEP .005,40
 880 LET A(B)=INT (RND*3)+1
 890 NEXT B
 900 GO SUB 660
 910 NEXT Z
 920 DIM Q(4)
 930 IF A(1)=A(2) AND A(2)=A(3) AND A(3)=A(4) THEN  L
ET Q$=' EL ESPECIAL!! ': LET PR=100: GO TO 960
 940 IF (A(1)=A(3) AND A(3)=A(4)) OR (A(1)=A(2) AND A
(2)=A(3)) OR (A(2)=A(3) AND A(3)=A(4)) OR (A(1)=A(2)
AND A(2)=A(4)) THEN  LET Q$=' 3 IGUALES!! ': LET PR=3
5
 950 IF A(1)=2 AND A(2)=2 OR A(2)=2 AND A(3)=2 OR A(3
)=2 AND A(4)=2 THEN  LET P$=' BONOS!! ': LET PR=PR+15
 960 FOR G=1 TO 15
 970 BEEP .008,50-G: BEEP .008,50
 980 NEXT G
 990 LET DIN=DIN+PR
1000 IF Q$='' THEN  GO TO 1070
1010 PRINT AT 19,6; BRIGHT 1; FLASH 1; INK 2; PAPER 4
;Q$
1020 PRINT TAB 6; BRIGHT 1; FLASH 1; INVERSE 1; INK 2
; PAPER 4;Q$
1030 IF PR>0 THEN  PRINT AT 10,4; BRIGHT 1; INK 2;'GA
NAS $';PR
1040 IF P$='' THEN  GO TO 1070
1050 PRINT AT 10,20; FLASH 1; INK 5; PAPER 3;P$
1060 PRINT AT 11,20; INVERSE 1; FLASH 1; INK 3;P$
1070 IF PR=0 THEN  GO TO 1170
1080 LET PP=0
1090 LET BOR=INT (RND*6)+2: FOR N=1 TO 50
1100 BEEP .01,40: BORDER BOR: BORDER 0
1110 NEXT N
1120 BORDER 0
1130 PRINT AT 2,28; FLASH 1; INVERSE 1; PAPER RND*4;
INK 9;DIN
1140 PRINT AT 19,6;'                ';TAB 6;'
     '
1150 PRINT AT 10,2; INK 6;'8888888888888'
1160 PRINT AT 10,20;'            ';AT 11,20;'
     '
1170 IF DIN<15 AND PR=0 THEN  PRINT AT 21,1; FLASH 1;
' NO TIENES DINERO PARA JUGAR ': PRINT #0;'  PULSA UN
A TECLA PARA EMPEZAR': GO TO 1190
1180 GO TO 1210
1190 IF INKEY$='' THEN  GO TO 1190
1200 RUN
1210 IF DIN>9500 THEN  PRINT AT 20,0; PAPER 2; INK 9;
 FLASH 1;'      ACABAS  DE  FUNDIRME!!     ': PRINT #0
;' PULSA UNA TECLA PARA COMENZAR ': GO TO 1230
1220 RETURN
1230 IF INKEY$='' THEN  GO TO 1230
1240 RUN
1250 REM [INV.] GRAFICOS USUARIO [TRUE]
1260 FOR Z=0 TO 7
1270 READ A,B,C,D
1280 POKE USR 'A'+Z,A
1290 POKE USR 'B'+Z,B
1300 POKE USR 'C'+Z,C
1310 POKE USR 'D'+Z,D
1320 NEXT Z
1330 LET A$(1)=CHR$ 16+CHR$ 2+'A'+'B'
1340 LET B$(1)=CHR$ 16+CHR$ 2+'C'+'D'
1350 FOR Z=0 TO 7
1360 READ E,F,G,H
1370 POKE USR 'E'+Z,E
1380 POKE USR 'F'+Z,F
1390 POKE USR 'G'+Z,G
1400 POKE USR 'H'+Z,H
1410 NEXT Z
1420 LET A$(2)=CHR$ 16+CHR$ 4+'E'+'F'
1430 LET B$(2)=CHR$ 16+CHR$ 4+'G'+'H'
1440 FOR Z=0 TO 7
1450 READ J,K,L,M
1460 POKE USR 'J'+Z,J
1470 POKE USR 'K'+Z,K
1480 POKE USR 'L'+Z,L
1490 POKE USR 'M'+Z,M
1500 NEXT Z
1510 LET A$(3)=CHR$ 16+CHR$ 3+'J'+'K'
1520 LET B$(3)=CHR$ 16+CHR$ 3+'L'+'M'
1530 RETURN
1540 DATA 0,8,121,254,0,28,63,254,0,48,63,254,15,32,3
1,254,31,240,15,252,51,252,7,248,52,254,1,224,101,254
,0,0,0,128,7,240,0,128,15,248,1,192,15,248,3,224,31,2
52,3,224,0,128,3,224,1,192,3,224,1,192,3,224,0,0,0,0,
1,136,0,2,26,8,56,12,60,16,124,52,126,208,127,204,61,
224,124,20,25,240,124,36,1,224,56,72,0,224
```

PASCAL

El PASCAL es actualmente el más firme candidato para desbancar al BASIC del lugar que ocupa dentro de los «micros», puesto que por su potencia, el PASCAL no sólo es apropiado para aprender informática sino para adentrarnos en cualquier campo sin caer en los defectos del BASIC.

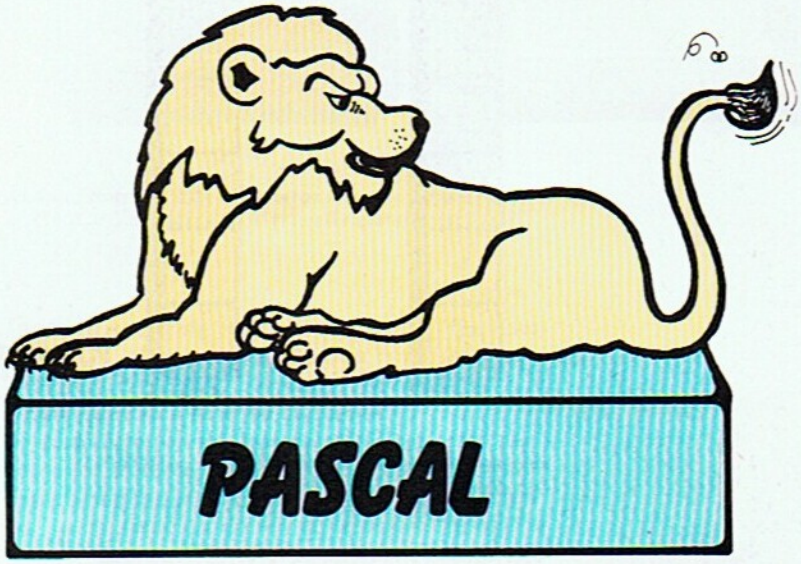
¿Por qué PASCAL? Este lenguaje surgió a finales de los años sesenta en un momento en que los ya existentes presentaban la particularidad que cuanto más largos eran los programas, más difícil era realizar un seguimiento visual de lo que hacen además de los problemas para realizar correcciones posteriores. En el caso particular del BASIC, el uso y abuso de la sentencia **GO TO** hace que para analizar un programa estemos saltando continuamente de un punto a otro disminuyendo bastante la inteligibilidad, pero que al ser un comando muy cómodo es fácil abusar de él.

El PASCAL soluciona estos problemas. Debido a su modularidad y a la posibilidad de trabajar con procedimientos, facilita bastante la comprensión y la corrección posterior de los programas.

Otro factor interviniente en el éxito que está teniendo el PASCAL, además de los ya mencionados, es que ocupa poca memoria y se puede incorporar en todos los microordenadores. Así ocurre con el ZX Spectrum que dispone de su propia versión PASCAL realizada por *HISOFT*. Es una versión que presenta como inconvenientes principales, además de tener la guía en inglés (pero que no sirve para nada a no ser que se conozca previamente el lenguaje), no presentar uno de los comandos para ficheros **(FILE)** que le da una gran potencia para el manejo de la información. No obstante, esta ausencia está motivada por una imposibilidad física, dado que no es posible soportarla en cinta, se hace imprescindible la utilización de un disco.

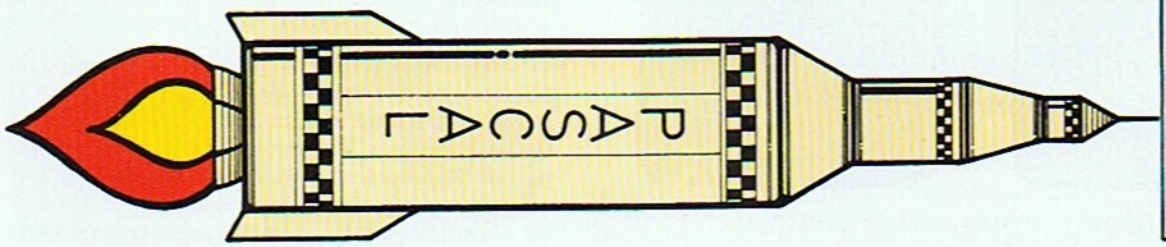
No debemos olvidar que el PASCAL es un lenguaje compilado, y que aparte de lo exigentes que son los compiladores (aunque en este caso es fácil acostumbrarse a ello) aporta la ventaja de su elevada velocidad, en ocasiones cuatro veces más rápido que el BASIC, lo cual hace su uso rentable para casi cualquier aplicación.

Una última advertencia: no nos desesperemos. El porqué se encuentra en que el BASIC crea unos vicios de programación (el abuso de **GO TO** y la no estructuración de los programas) que son incompatibles con la filosofía del PASCAL. Le costará tiempo y programas sacar el máximo provecho a este lenguaje pero una vez que lo consiga se dará cuenta de lo ventajoso que resulta. Debe tener en cuenta que el PASCAL ha sido creado



El PASCAL debido a su potencia y modularidad es uno de los más firmes candidatos para desplazar al BASIC en los microordenadores.

Al ser un compilador, el PASCAL es más rápido que el BASIC.

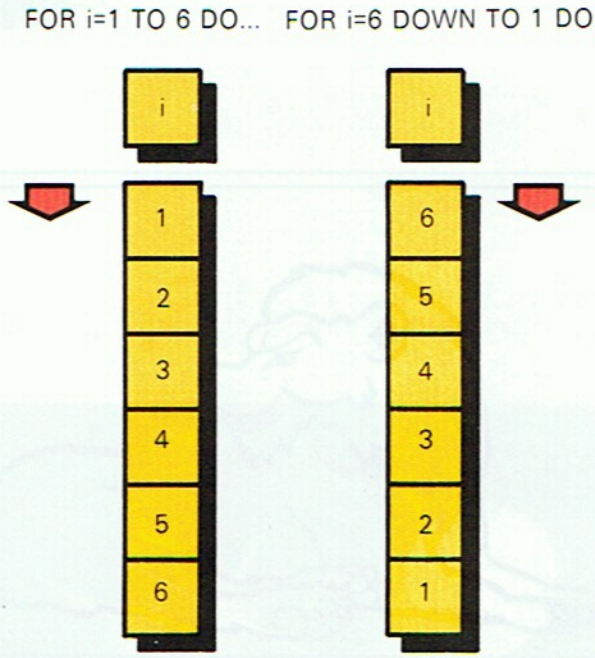


¡!

Antes de empezar a teclear un programa PASCAL hay que tener muy claras las ideas y un esquema de lo que pretendemos.

*

El PASCAL surgió para solucionar los problemas que presentaban los lenguajes de la primera época informática: dificultad de seguir un programa largo y de realizar correcciones posteriores.



Con FOR ... TO ... DO *la variable de control del bucle va aumentando en una unidad, con* FOR ... DOWNTO ... DO *disminuye en una unidad.*

para que lo comprenda el usuario y no la máquina y esto se nota cuando trabajamos con él.

LOS PROGRAMAS

Quizá pueda parecer empezar la casa por el tejado el explicar primeramente la estructura de los programas sin conocer antes algunos comandos y su función. Sin embargo, por la estructura del PASCAL, al ser un compilador, es imposible introducir comandos directos, como ocurre con el BASIC. Todos los comandos deben ser introducidos formando parte de un programa y, por tanto, empleando su sintaxis y estructuración.

Todo programa PASCAL estará formado por las siguientes partes:

1. Asignación de un nombre al programa. Es lo primero a realizar y se utiliza el comando **PROGRAM** seguido del nombre.

10 PROGRAM sumayresta;

En el ejemplo se da el nombre "sumayresta" a un programa.

2. Especificación de los parámetros del programa. Estos pueden ser **INPUT** u **OUTPUT**, indicando que el programa realizará operaciones de entrada y salida, respectivamente. Se indica de la siguiente manera:

PROGRAM sumayresta (INPUT,OUTPUT);

Sin embargo, en el *HISOFT PASCAL* este tipo de indicación no sólo no es imprescindible sino que no es admitida, debido a las características propias de esta versión. Es necesario después de cada línea poner el «;».

3. Ahora llega la parte de declaración, en donde se enuncian primeramente el grupo de constantes (indicado con **CONST**) y el de variables (indicado con **VAR**) que serán utilizados a lo largo de todo el programa.

```
20 CONST
30     numa=3;
40     numb=2;
50 VAR
60     ressuma,resresta:INTEGER;
```

En esta ocasión se inicializan las constantes «base» y «suma» a «2» y «3» respectivamente. Por otro lado, se definen unas variables («exponente» y «resultado») que van a tomar valores enteros **(INTEGER)**, las variables también pueden tomar valores reales **(REAL)**.

Además de declarar constantes y variables en este apartado también son declarados otros parámetros que más adelante veremos.

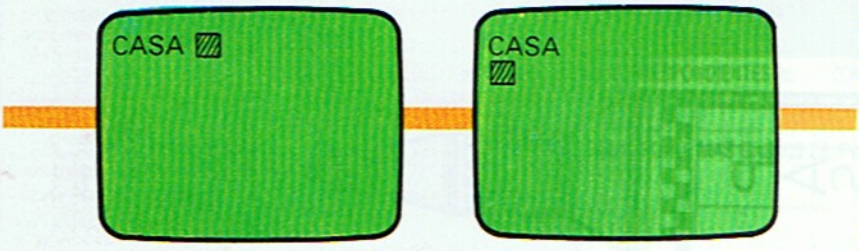
4. Aquí es donde entra el programa en sí. Se encabeza por la palabra **BEGIN** y se finaliza con **END.** Un punto («.») a continuación de esta palabra indica que es el final del texto del programa.

```
 70 BEGIN
 80     ressuma:=numa+numb;   halla la suma
 90     resresta:=numa-numb;  halla la resta
100     WRITE (ressuma,resresta);  imprime resultado
110 END.
```

5. Sólo queda por mencionar los comentarios de ayuda al programador. No es un apartado en sí pero no puede ser introducido en ninguna de las partes anteriores. Se pueden colocar en cualquier punto del programa y, en la versión de HISOFT, debe estar encerrado entre corchetes.

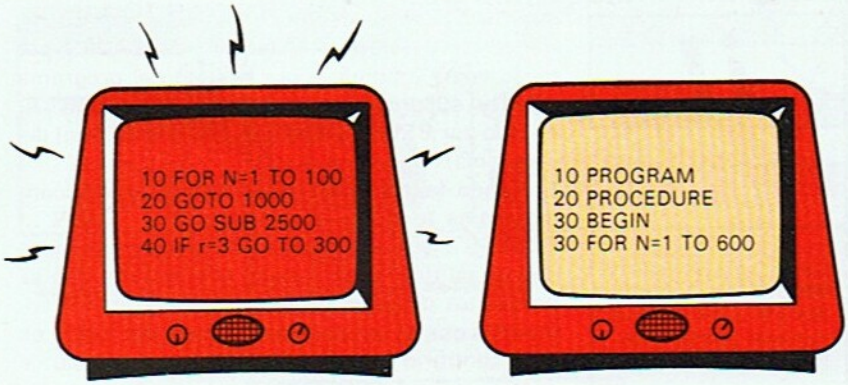
Es muy común a la hora de programar en PAS-

El comando WRITE *mantiene el cursor en la misma línea de impresión y* WRITELN *salta a la línea siguiente.*



WRITE WRITELN

CAL no situar todos los comandos sobre una misma columna, unos encima de otros, sino irlos introduciendo dentro de la línea según vamos avanzando en la programación. Así, en el ejemplo anterior **BEGIN** sobresale sobre lo escrito en las líneas inferiores. Esto es lo que se conoce por formato libre, y es aconsejable utilizarlo para una mejor comprensión de los programas.
Aunque en la versión de HISOFT PASCAL aparezcan los números de líneas, éstos sólo sirven para facilitar la edición y corrección de ellas no influyendo en la programación. De hecho, algunas versiones de otros ordenadores no incorporan los números.



Al no utilizar **GO TO**, *entre otros factores, un programa en PASCAL es más fácil de entender que uno en BASIC.*

ESCRIBIMOS Y VEMOS

Ya en el ejemplo anterior vimos una sentencia de visualización por pantalla del resultado *(WRITE)*. Si queremos una presentación mejor podemos hacer la siguiente modificación:

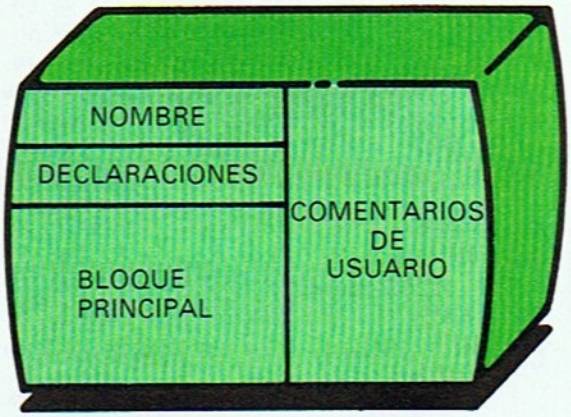
WRITELN ("SUMA=",ressuma);
WRITE ("RESTA=",resresta);

WRITELN escribe lo que tiene entre paréntesis pero al finalizar salta el cursor a la línea siguiente, al contrario de **WRITE** que deja el cursor a continuación del último literal representado.
Para la introducción de los datos disponemos del comando **READ** que permite la lectura de un dato por teclado y lo asigna a una variable.

READ (numa,numb)

En este caso, el programa espera que introduzcamos dos datos separados por unos espacios. Es fácil observar que el primer parámetro se asignará a "numa" y el segundo a "numb".

En la programación del HISOFT-PASCAL hay que tener en cuenta los bloques que forman un programa.



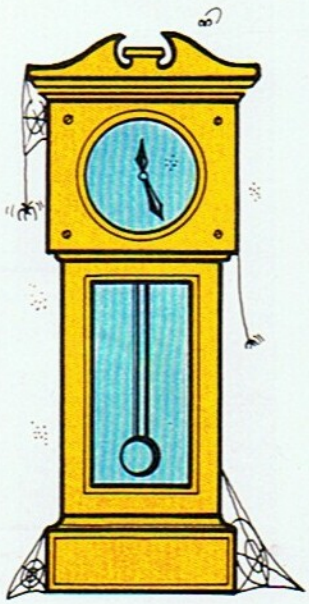
También podemos provocar el salto de línea con **READLN** (lógico ¿verdad?), que como se podrá imaginar, después de la introducción del dato el cursor cambia a la línea siguiente.
Si queremos introducir las modificaciones presentadas en este apartado al programa anterior, deberemos tener en cuenta que hemos de eliminar la declaración de las constantes "numa" y "numb" (porque los valores se introducen ahora por teclado y no tendrán siempre el mismo valor) y cambiarlas a variables de tipo entero, situándolas a continuación de "resresta".

LOS BUCLES Y LAS SENTENCIAS DE CONTROL

El PASCAL dispone de un amplio repertorio de sentencias que puedan entrar dentro de este grupo. Vamos a conocerlas empezando por las más fáciles.
Como bucle propiamente dicho tenemos las sentencias **FOR...TO...DO** y **FOR...DOWNTO...DO** que son totalmente equivalentes al **"FOR...TO"** del BASIC. La primera de ellas incrementa el valor de una variable en una unidad hasta que se alcanza el número deseado. La segunda es prácticamente igual, pero provoca un decremento en una unidad. El PASCAL no admite saltos diferentes a la unidad.

FOR i:=1 TO 6 DO ...

En este caso, a la variable "i" se le asigna el valor inicial "1", y el valor máximo que tomará será "6", ejecutando todos los comandos que se en-



¡!

Como características destaca su modularidad y su posibilidad de manejar procedimientos.

*

Por sus características ocupa poca memoria, debido a lo cual puede ser incorporado en la mayoría de los microordenadores.

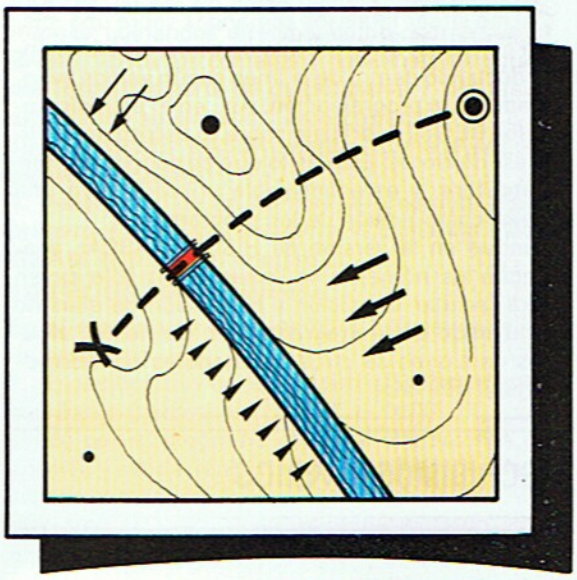
cuentren después de "DO". No existe orden de finalización del bucle (el **"NEXT"** del BASIC), por tanto viene limitado o por el final del programa o por un subprograma situado a continuación y limitado por **BEGIN** y **END;** (o **END.** si es final del programa).

Entrando dentro de las sentencias de control condicionales tenemos el conocido **IF ... THEN ... ELSE ...** , que no presenta ninguna particularidad en su uso: si se cumple la condición que se encuentra después de **IF** se ejecutarán los comandos que se encuentran después del **THEN** en caso contrario saltará y el programa continuará a continuación del **ELSE,** cuya colocación es opcional.

En el caso de menús es necesario incluir una sentencia **IF** por cada opción para en cada caso realizar la acción deseada. En PASCAL nos podemos ahorrar todos los condicionales por una sola instrucción: **CASE ... OF.** Así, si sabemos que una variable puede tomar diferentes valores (por ejemplo 6, 9 y 50) y dependiendo de cual sea hay que realizar unas acciones diferentes el problema se soluciona así:

```
CASE fondo OF
6:fondo:=0;
9:WRITE (fondo);
50:lado:=lado*5;
```

Examinamos el valor de la variable "fondo". Si vale "6" se le asigna a ella "0", si toma el valor "9" se imprime su valor y si es "50" a una variable que se utiliza en el programa ("dato") se le multiplica por "5" y se le asigna a ella misma. Una sentencia incorporada en muchos BASIC de los «micros» actuales pero que el Spectrum no «conoce» es **WHILE ... DO ...** Después de **WHILE** se inserta una condición y las sentencias que se encuentran después de **DO** se ejecutan continuamente mientras la condición sea cierta.



Con el HISOFT-PASCAL podemos realizar dibujos al llevar incorporados los gráficos de tortuga, similares a los del LOGO.

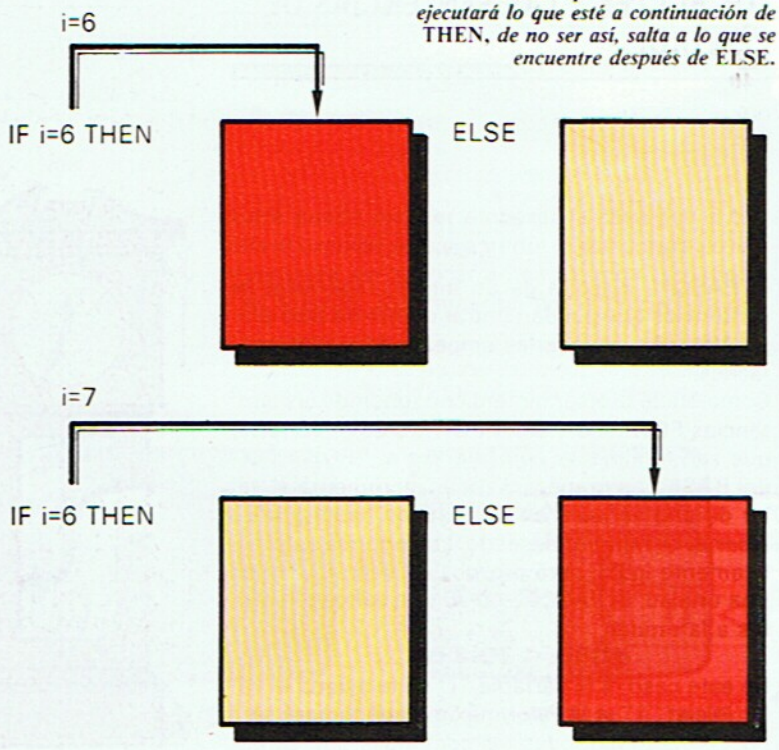
```
   ...
50 READ (num);
60 WHILE num<6 DO
70   BEGIN;
80     READ (num);
90   END;
   ...
```

En este fragmento del programa, primeramente se nos pide un número que le asignamos a "num", y mientras "num" vaya valiendo menos de "6" se ejecuta el subprograma a continuación de **DO** el cual nos vuelve a pedir otra introducción por teclado.

La última instrucción condicional que presenta el PASCAL es **REPEAT ... UNTIL,** en la cual se ejecutan las sentencias que se encuentra entre ambas palabras mientras la condición que se encuentra después de **UNTIL** sea falsa. Es decir, primero se ejecuta una vez y luego se realiza la pregunta sobre si continúa o no. Si ahora queremos que el fragmento del programa anterior haga lo mismo pero con estos comandos nuevos deberemos efectuar:

```
   ...
50 REPEAT;
60   BEGIN;
70     READ (num);
80   END;
90 UNTIL num>=6; {teclear ">" y "=", no
">="!!!}
   ...
```

Si se cumple la condición IF *se ejecutará lo que esté a continuación de* THEN, *de no ser así, salta a lo que se encuentre después de* ELSE.

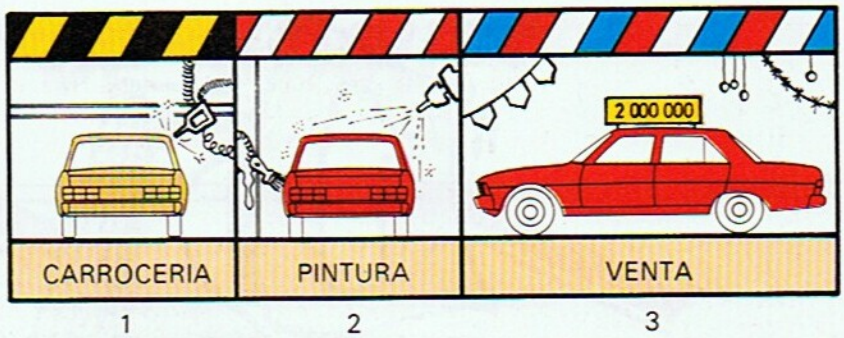


Si "num" es menor que "6" el bucle se volverá a ejecutar, es decir, provoca el efecto contrario a **WHILE ... DO.**

Siempre que una variable va a ser utilizada solamente en una sentencia condicional es preciso declararla antes como **"BOOLEAN"** para indicar que va a ser empleada en una comparación. En el ejemplo anterior, si "num" no se utilizara en la introducción por teclado de un valor, y por tanto, se declarara como **"INTEGER"**, habría que efectuar:

VAR num:BOOLEAN;

El PASCAL también conoce la sentencia **GO TO** pero se aconseja utilizarla lo menos posible. El **GO TO** es un salto a un número, llamado etiqueta, que se declara previamente **(LABEL 100)**, nunca el salto será a un número de línea.



Un procedimiento es un fragmento de programa que realiza una sola función.

LOS PROCEDIMIENTOS

Los procedimientos y las funciones (una variante de aquellos) son equivalentes, aunque con mayor potencia, a las subrutinas del BASIC llamadas por **GOSUB.** Es la pieza fundamental en la realización de un programa en PASCAL.

Supongamos que tenemos entre manos un programa que realiza una serie de operaciones diferentes, lo ideal sería dividirlo en pequeños fragmentos que hicieran por separado una sola función. Cada uno de estos fragmentos es un procedimiento.

Vamos a hacer un programa que calculará el cuadrado de un número y para ello utilizaremos un procedimiento que será el encargado de realizar la operación. Para ello, después de realizados los preliminares que ya conocemos, debemos declarar el procedimiento y escribir los comandos para que realice nuestra función. Dentro del procedimiento debemos declarar todas las variables, constantes y parámetros que se utilicen, como si de un programa más se tratase.

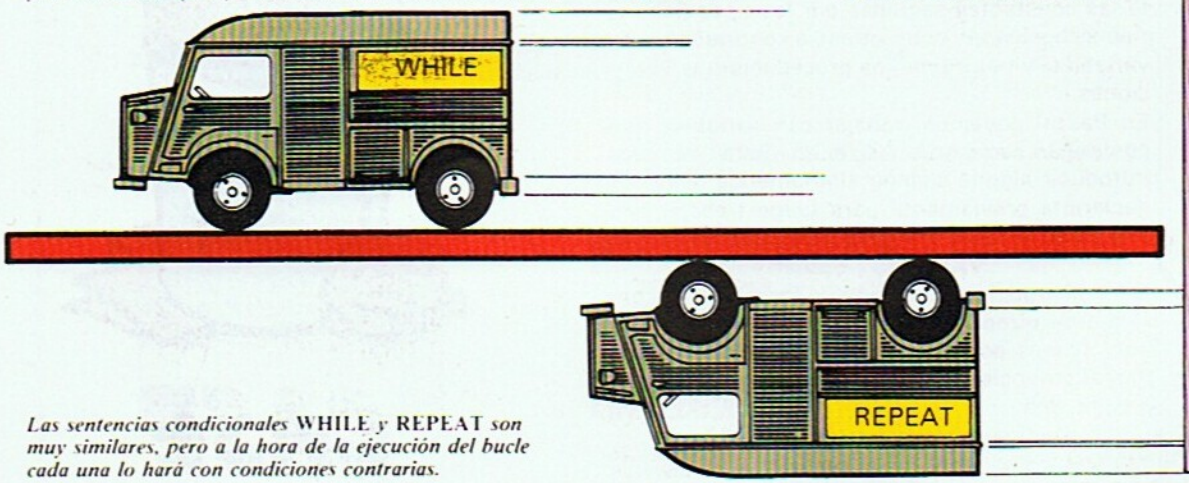
```
10 PROGRAM cuadrado;
20 VAR numero:INTEGER;
30    PROCEDURE potencia; {Declaración procedimiento}
40       VAR resul:INTEGER; {Comienza el procedimiento}
50       BEGIN
60          resul:=numero*numero;
70          numero:=resul;
80       END;          {Fin del procedimiento}
90 BEGIN               {Comienzo del programa}
```

¡!

Es un lenguaje que ha sido pensado para que el usuario lo entienda fácilmente, y no para que lo entienda la máquina solamente.

*

El PASCAL es un lenguaje compilado y, aparte de lo exigentes que son los compiladores, aporta la ventaja de su elevada velocidad.

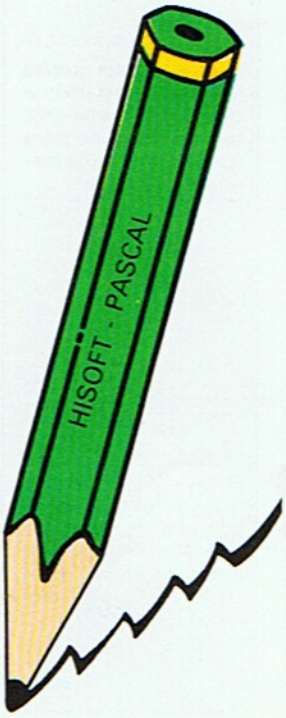


Las sentencias condicionales WHILE *y* REPEAT *son muy similares, pero a la hora de la ejecución del bucle cada una lo hará con condiciones contrarias.*

A la hora de realizar las declaraciones debe seguirse siempre el mismo orden.

```
100    READ (numero);
110    potencia;        {Llamada al procedimiento}
120    WRITE (numero);
130  END.               {Fin del programa}
```

LAS DECLARACIONES



Aunque se ha estado mencionando a lo largo del capítulo, quizá convenga realizar un agrupamiento de todo lo mencionado.

Es necesario que las declaraciones sigan un orden para que todo marche bien; así, primero serán las etiquetas (para los **GO TO**) posteriormente las constantes seguidas por los tipos (las variables booleanas entre otras), a continuación las variables, y por último los procedimientos y funciones.

En Pascal podemos trabajar con variables que contengan caracteres. Así, si en "letra" vamos a introducir alguna cadena alfanumérica debemos declararla previamente para luego trabajar con ella como si fuera una más.

VAR letra : CHAR

Sin embargo, la declaración de los diferentes parámetros ofrece mucha más libertad de lo que aquí hemos podido ver, dando al PASCAL una mayor potencia en la programación. He aquí una simple idea:

medioanno=(enero,febrero,marzo, abril,mayo,junio)

Podremos trabajar con la variable "medioanno" como si fuera casi un número. Observemos lo siguiente:

FOR mes=enero TO junio DO ...

Los valores de la variable "mes" variarán desde "enero" hasta "junio", ¡como si fuera un bucle con números!

LLEGO EL FIN

Esperamos que esta pequeña revisión haya sido suficiente, aunque tan solo sea para vislumbrar la potencia que puede aportar el PASCAL en cuanto se domine.

En las versiones estándar se hechan de menos los comandos gráficos (puesto que la finalidad inicial del lenguaje no era ésta y suponía un lujo para la época). Sin embargo, con el HISOFT tenemos esta posibilidad (además del sonido), que se hace casi obligatoria en los «micros» actuales, poniéndose así a nuestro alcance cualquier aspecto que queramos programar, sobre todo teniendo en cuenta que nos ofrece los gráficos de tortuga, que son totalmente similares a los del LOGO.

Un último consejo: el PASCAL es un lenguaje para hacerse un esquema antes de empezar a teclear en la máquina y tener las ideas muy claras. No así en el BASIC, donde directamente podemos empezar a teclear el programa.

La función CASE nos supone un ahorro de sentencias IF.

MOVIMIENTOS DE BLOQUES

En las próximas líneas discutiremos tres nuevos grupos de instrucciones denominados en las tablas del capítulo inicial de esta serie de intercambio, transferencia y búsqueda de bloques, a los cuales añadiremos antes de finalizar, el grupo lógico de 8 bits. Comencemos sin más, por el primero de ellos.

EL GRUPO DE INTERCAMBIO

Las instrucciones que lo componen provocan el intercambio de la información contenida en los registros implicados. No tiene sentido hablar de origen y destino, pues el intercambio es biyectivo. Consideremos la primera de ellas: **EX DE,HL** (el mnemónico EX responde a la abreviatura de la palabra inglesa *EXchange,* canje o intercambio). Mediante ella, el valor almacenado en el par DE se transfiere al par HL, y simultáneamente, el contenido de HL pasa a DE.

Siguiendo el mismo mecanismo, tres instrucciones se encargan de intercambiar información con el stack: **EX(SP),HL EX(SP),IX EX(SP),IY,** de manera tal, que permiten canjear el valor actual de los registros HL, IX o IY con la última entrada efectuada en la pila. Y lo que es más importante: el contenido del Stack Pointer SP no resulta alterado.

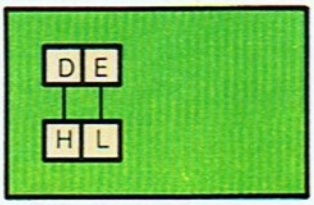
LOS REGISTROS ALTERNATIVOS

Las instrucciones de intercambio habilitan el uso del juego de registros alternativos, el cual se utiliza habitualmente en C/M para conservar valores o direcciones de interés que requieran una fácil y rápida recuperación, a la par de seguridad ante una posible corrupción como puede ocurrir, por ejemplo, en el stack.

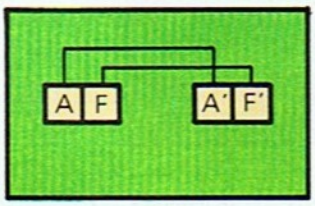
EX AF,A'F', efectúa el canje de los valores contenidos en los pares AF y A'F'. Finalmente, completa el grupo de intercambio una potente ins-

ENSAMBLADOR	CODIGO MAQUINA	Nº BYTES	CICLOS RELOJ	CICLOS MAQUINA
EX DE,HL	1 1 1 0 1 0 1 1	1	4	1
EX AF,AF'	0 0 0 0 1 0 0 0	1	4	1

INDICADORES: NO AFECTADOS



Z 80



Z 80

Al ejecutar la instrucción OR s, se efectúa la suma lógica entre los bits correspondientes de los dos octetos implicados.

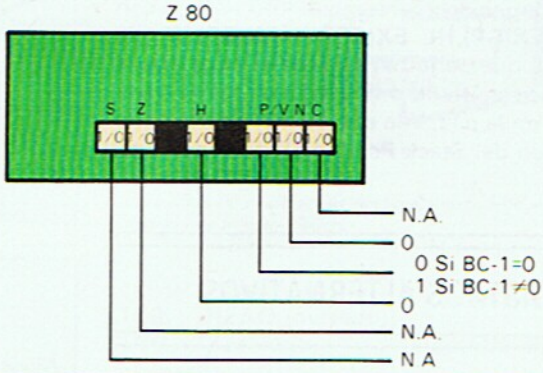
*

Las instrucciones **LDI, LDD, CPI, CPD** carecen de autorrepetición.

*

Si deseamos examinar un área determinada de memoria para saber si un valor particular está presente en ella, situaremos el origen en el par HI, el número de bits a comparar en BC, y en el acumulador A el valor a ser contrastado.

ENSAMBLADOR	CODIGO MAQUINA	Nº BYTES	CICLOS RELOJ	CICLOS MAQUINA
LDI	11101101 10100000	2	16	4
LDD	11101101 10101000	2	16	4

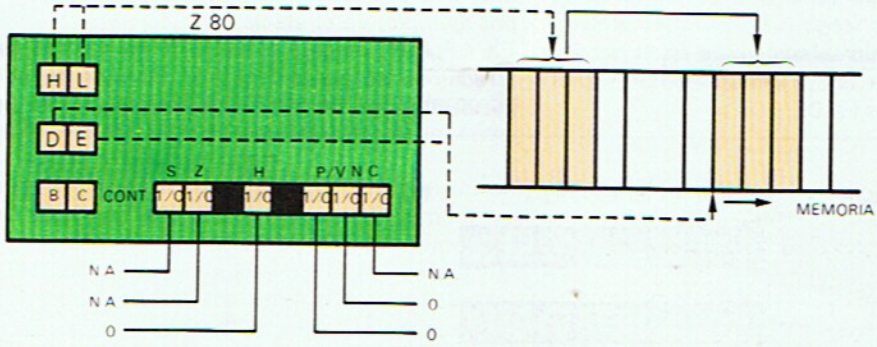


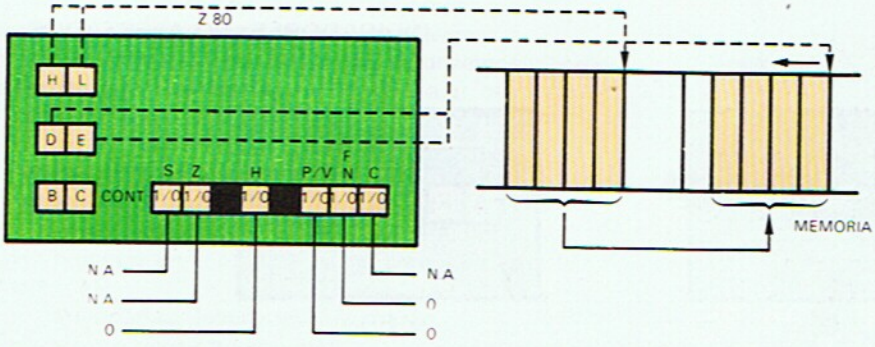
trucción: **EXX**, la cual de una sola vez intercambia los datos almacenados en los registros BC, DE y HL con los contenidos de B'C', D'E' y H'L', respectivamente.

MANEJANDO BLOQUES

Entre las instrucciones desarrolladas para el microprocesador Z80, ocho fueron las previstas para gestionar la transferencia y búsqueda de bloques. Las primeras permiten al programador trasladar un área de memoria de una zona a otra de ésta. Las segundas examinan si una determinada configuración se haya presente dentro de un bloque de datos.

ENSAMBLADOR	CODIGO MAQUINA	Nº BYTES	CICLOS RELOJ	CICLOS MAQUINA
LDIR	11101101 10110000	2	21 BC≠0 16 BC=0	5 4
LDDR	11101101 10111000	2	21 BC≠0 16 BC=0	5 4





¡!

Las tres instrucciones que se encargan de intercambiar información con el stack son: **EX(SP),HL, EX(SP),IX, EX(SP),IY.**

*

Entre las instrucciones desarrolladas para el microprocesador Z80, ocho fueron previstas para gestionar la transferencia y búsqueda de bloques.

Cuando efectuemos la transferencia de un bloque, la dirección de origen de éste debemos cargarla previamente en el par HL, y la dirección de destino, en el par DE. Por su parte, los registros BC han de contener el número de bytes a ser transferidos.

Si nuestro objetivo es examinar un área determinada de memoria al objeto de confirmar o no, si un valor particular está presente en ella colocaremos el origen de ésta en el par HL, el número de bytes a comparar en BC, y en el acumulador A, el valor particular a ser contrastado.

Podemos clasificarlas en dos subgrupos:

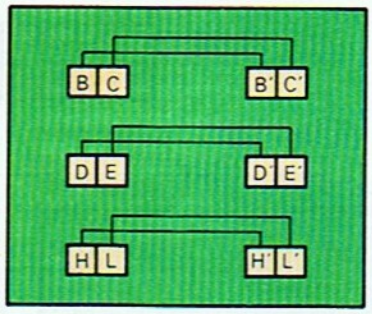
1) Funcionamiento automático **(LDIR, LDDR, CPIR, CPDR).**

Estas instrucciones ejecutan cíclicamente su trabajo hasta que una determinada condición les indica que ha finalizado éste. Por ejemplo, supongamos que deseamos transferir un bloque de una zona a otra mediante la orden LDIR (LD-cargar, I-incrementar, R-repetir). Internamente, el microprocesador efectúa las siguientes operaciones:

— T1: Carga el contenido de la posición de memoria direccionada por DE (como sabemos se tra-

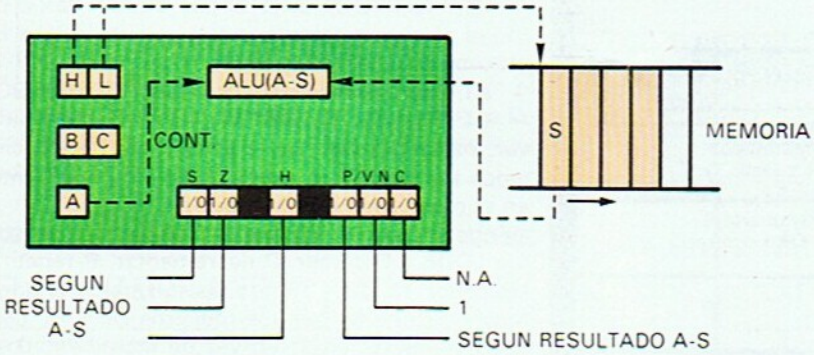
ENSAMBLADOR	CODIGO MAQUINA	Nº BYTES	CILOS RELOJ	CICLOS MAQUINA
EXX	1 1 0 1 1 0 0 1	1	4	1

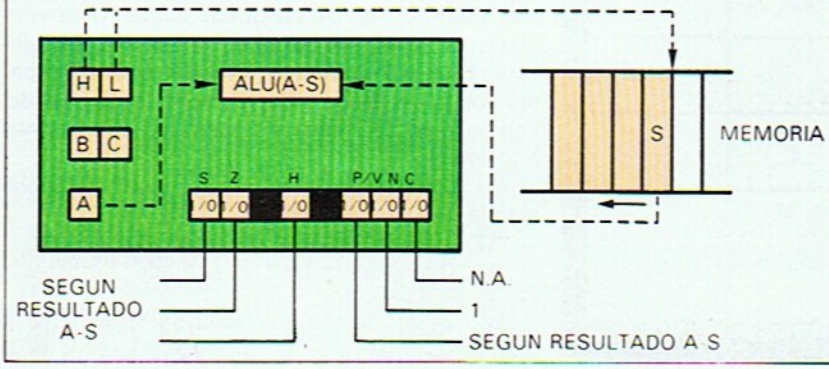
INDICADORES: NO AFECTADOS



Z 80

ENSAMBLADOR	CODIGO MAQUINA	Nº BYTES	CICLOS RELOJ	CICLOS MAQUINA
CPIR	1 1 1 0 1 1 0 1 1 0 1 1 0 0 0 1	2	21 BC≠0 y A≠(HL) 16 BC=0 ó A=(HL)	5 4
CPDR	1 1 1 0 1 1 0 1 1 0 1 1 1 0 0 1	2	21 BC≠0 y A≠(HL) 16 BC=0 ó A=(HL)	5 4



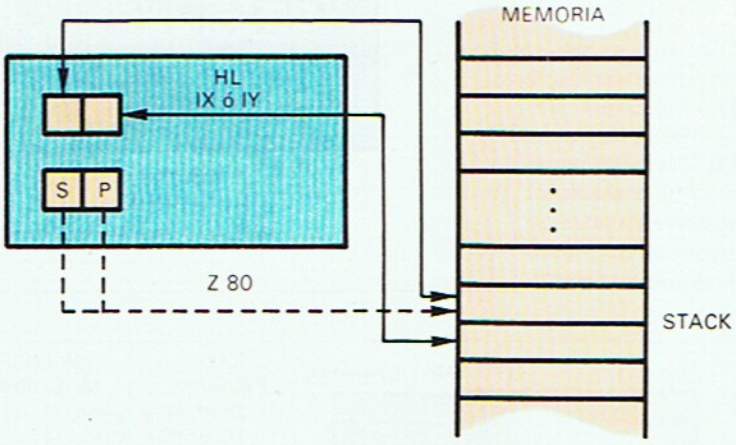


Las instrucciones que componen el grupo de intercambio provocan el intercambio de la información contenida en los registros.

Las instrucciones **LDIR, LDDR, CPIR, CPDR** ejecutan cíclicamente su trabajo hasta que una determinada condición les indica que ha finalizado éste.

ENSAMBLADOR	CODIGO MAQUINA	Nº BYTES	CICLOS RELOJ	CICLOS MAQUINA
EX (SP),HL	1 1 1 0 0 0 1 1	1	19	5
EX (SP),IX	1 1 0 1 1 1 0 1 1 1 1 0 0 0 1 1	2	23	6
EX (SP),IY	1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 1	2	23	6

INDICADORES: NO AFECTADOS



OR s				
OPERANDO	HEX.	DEC.	INDICADOR	
n	F6,n	246,n	S	1 si resultado – 0 si resultado ++
A	B7	183	H	1 si resultado 0 0 en caso contrario
B	B0	176		
C	B1	177	H	0
D	B2	178	P/V	1 paridad par 0 paridad impar
E	B3	179		
H	B4	180	N	0
L	B5	181	C	0
(HL)	B6	182		
(IX+d)	DD,B6,d	221,182,d	S OPERANDO	
(IY+d)	FD,B6,d	253,182,d		

ta de la dirección destino) con el octeto señalado por HL (dirección origen o fuente).

— T2: Incrementa en 1 el par DE, así como el HL, al objeto de transferir el siguiente dato a la próxima posición de destino.

— T3: Decrementa en 1 el contador de bytes BC.

— T4: El mismo proceso es ejecutado una y otra vez, hasta que por fin, el par BC es 0, es decir, todos los datos han sido transferidos, momento en el cual termina la instrucción.

Idénticas consideraciones es factible efectuar sobre LDDR (LD-cargar, D-decrementar, R-repetir): los pasos T1, T3 y T4 son los mismos y la diferencia estriba en que los registros DE y HL son decrementados en 1, en vez de incrementados, en el paso T2.

Las instrucciones de búsqueda actúan de manera similar. Sea, por ejemplo, nuestro objeto ejecutar CPIR (CP-comparar, I-incrementar, R-repetir) sobre un bloque de memoria determinado. Para ello el Z80 llevará a cabo las operaciones siguientes:

— B1: Compara el contenido del acumulador con el byte direccionado por HL.

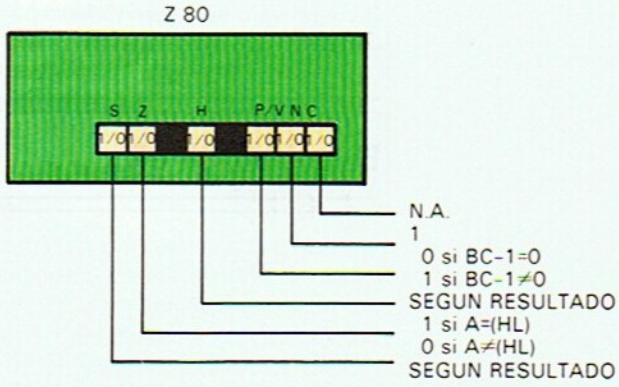
— B2: Incrementa en 1 el par de registros HL.

— B3: Decrementa en 1 el contador de bytes a ser examinados BC.

— B4: Repite el proceso anterior hasta que, o bien se encuentra en la zona escrutada un valor

idéntico al del acumulador, o bien, no quedan más bytes a ser comparados, circunstancia satisfecha cuando BC es 0.
CPDR (CP-comparar, D-decrementar, R-repetir) es a las instrucciones de búsqueda lo que LDDR es a las de transferencia de bloques. La única diferencia en cuanto a funcionamiento con CPIR reside en el paso B2, donde el par HL es decrementado en vez de incrementado en 1.
2) Mecanismo no automático **(LDI, LDD, CPI, CPD).**
Estas cuatro instrucciones tienen en común con las del grupo anterior los tres primeros pasos indicados, pero carecen de la autorrepetición. Por tanto, al construir un programa que las implemente deberemos hacer uso de ellas, una vez por cada byte a ser transferido o comparado.

ENSAMBLADOR	CODIGO MAQUINA	Nº BYTES	CICLOS RELOJ	CICLOS MAQUINA
CPI	1 1 1 0 1 1 0 1 1 0 1 0 0 0 0 1	2	16	4
CPD	1 1 1 0 1 1 0 1 1 0 1 0 1 0 0 1	2	16	4



LAS INSTRUCCIONES LOGICAS

Llegado este apartado os recomendamos antes de proseguir con él repasar el capítulo dedicado al Algebra de Boole, punto de partida de todo lo que seguirá a continuación.
Las instrucciones lógicas enfrentan dos octetos entre sí efectuando la operación (AND, OR, XOR) entre sus bits individuales tomándolos por parejas formadas por aquellos que ocupan el mismo lugar dentro de los bytes considerados.
El primero de ellos ha de estar contenido en el acumulador A, y el segundo en un registro o localización de memoria específica. El resultado siempre se recoge en A.

EL OPERADOR AND

En la figura está representada su tabla de verdad. Como observaremos, efectúa el producto lógico entre los bits implicados, es decir, el resultado es 1 si, y sólo si, los dos son 1, siendo 0 si al menos uno de ellos es 0.
Una de las operaciones más habituales en los programas en C/M consiste en aplicarlo sobre el mismo acumulador, es decir, implementar la ins-

AND s

OPERANDO	HEX.	DEC.	INDICADOR	
n	E6,n	230,n	S	1 si resultado – 0 si resultado +
A	A7	167	Z	1 si resultado 0 0 en caso contrario
B	A0	160		
C	A1	161	H	1
D	A2	162	P/V	1 paridad par 0 paridad impar
E	A3	163		
H	A4	164	N	0
L	A5	165	C	0
(HL)	A6	166		
(IX+d)	DD,A6,d	222,166,d	S OPERANDO	
(IY+d)	FD,A6,d	253,166,d		

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

XOR	0	1
0	0	1
1	1	0

trucción AND A. Con ello, el contenido de este registro se mantiene inalterado, pero modifica algunos bits del registro F, de tal manera, que es posible identificar si el valor almacenado en A es 0, negativo o tiene un número par de unos (paridad par).

EL OPERADOR OR

Al ejecutar la instrucción OR s, donde s es cualquier valor almacenado en los registros o posiciones de memoria que se detallan en las figuras, se efectúa la suma lógica entre los bits correspondientes de los dos octetos implicados.
Hemos de tener presente que en toda operación lógica no se producen acarreos, es decir, el resultado obtenido tras aplicar cualquiera de los operadores sobre una pareja de bits, no tiene ninguna influencia sobre los siguientes, siendo tratadas de forma independiente.
Al igual que antes analizando el contenido del registro F tras implementar la instrucción OR A, estamos en condiciones de precisar si el dato almacenado en el acumulador es negativo, cero o tiene paridad par.

OR EXCLUSIVO XOR

XOR s				
OPERANDO	HEX.	DEC.	INDICADOR	
n	EE,n	238,n	S	1 si resultado – 0 si resultado +
A	AF	175	Z	1 si resultado 0 0 en caso contrario
B	A8	168		
C	A9	169	H	0
D	AA	170	P/V	1 si paridad par 0 si paridad impar
E	AB	171		
H	AC	172	N	0
L	AD	173	C	0
(HL)	AE	174		
(IX+d)	DD,AE,d	221,174,d	S OPERANDO	
(IY+d)	FD,AE,d	253,174,d		

El resultado de este operador aplicado sobre dos octetos determinados conduce a la comparación lógica de ambos, es decir, cuando enfrentamos dos bits se obtiene 1 si son diferentes, y 0 en el caso de ser iguales.
Muchos programas utilizan la instrucción XOR A como sustituto de LD A,0, siempre y cuando no tenga importancia modificar el contenido del registro F, puesto que se consigue ahorrar un byte por dicho sistema.

MASCARAS

Una de las utilidades más importantes ligada al empleo de las instrucciones lógicas es la de permitir una técnica de programación basada en las máscaras (en inglés *mask*). En la tabla podemos encontrar algunos ejemplos relacionados con ellas.

1. Eliminar ciertos bits dentro de un byte (Ej. In seis superiores)

0	1	1	0	0	1	1	0	
0	0	0	0	0	0	1	1	MASCARA AND
0	0	0	0	0	0	1	0	RESULTADO

2. Paso de minúscula a mayúscula

0	1	0	0	0	0	0	1	A.S.C.I.I. de la «A»
0	0	1	0	0	0	0	0	MASCARA OR
0	1	1	0	0	0	0	1	RESULTADO (A.S.C.I.I. de la «a»)

Quizá fuera más preciso utilizar el término «filtro», pues el objetivo final de dicha técnica consiste en seleccionar ciertos bits de interés para nosotros dentro de un octeto, obviando o inhibiendo los restantes.

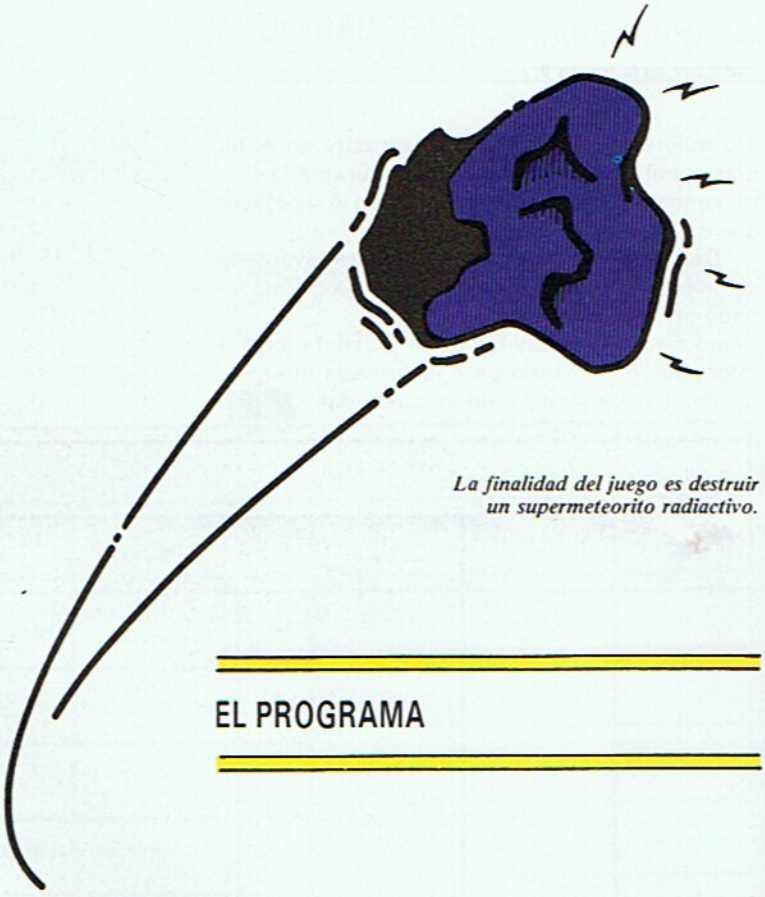
COMPARACIONES

Para completar el grupo lógico de 8 bits nos queda analizar las instrucciones de código mnemónico CP (*ComPare,* comparar), las cuales confrontan el contenido del acumulador con un byte s determinado por el valor almacenado en un registro o posición de memoria específica.
Como sabemos, el resultado de una comparación solamente ofrece dos posibilidades: o las dos cantidades son iguales o son diferentes. Cuando se trata del segundo caso puede interesarnos conocer cual es la mayor (o cual es la menor).
El Z80 soluciona la papeleta con habilidad ejecutando internamente la resta A-s y activando algunos indicadores los cuales delatan sin duda el resultado de la comparación, manteniendo inalterados los contenidos tanto del acumulador como del byte s. Comprobémoslo:
— Si la bandera de cero Z está alzada, es decir, contiene un 1, entonces la resta A-s fue 0, y por tanto, A y s son iguales.
— En caso contrario Z contendrá un 0. Nos fijamos ahora en el indicador de acarreo C. Si está a 0 quiere decir que no hubo arrastre en el bit más significativo del acumulador durante la resta, condición que se cumple cuando A es mayor que s.
— De no ser así, la bandera C estará levantada señalando que s es mayor que el valor almacenado en el acumulador.
Como siempre, en las tablas se presenta toda la información necesaria para la correcta interpretación de cada instrucción en particular.

CP s				
OPERANDO	HEX.	DEC.	INDICADORES	
n	FE,n	254,n	S	1 si resultado – 0 si resultado +
A	BF	191	Z	1 si resultado 0 0 en caso contrario
B	B8	184		
C	B9	185	H	1 si acarreo del BIT 3
D	BA	186	P/V	1 si hay sobrepasamiento
E	BB	187		
H	BC	188	N	1
L	BD	189	C	1 si hay acarreo
(HL)	BE	190		
(IX+d)	DD,BE,d	221,190,d	S OPERANDO	
(IY+d)	FD,BE,d	223,190,d		

RADAR

Nuestro monitor o televisor acaba de convertirse en una pantalla de radar de seguimiento. Podemos vigilar todos los movimientos de cualquier estrella del universo y, lo que es más importante: salvaguardar nuestro planeta de las oleadas de meteoritos radiactivos que últimamente surcan el espacio exterior cercano a la Tierra.



La finalidad del juego es destruir un supermeteorito radiactivo.

EL PROGRAMA

Una vez el programa ha sido debidamente introducido y ejecutado, aparecerá en la pantalla un menú de selección de dificultad. Dicho menú sólo posee dos opciones: fácil o difícil. La complejidad del juego dependerá del número de estrellas estáticas que surjan en el monitor. Lógicamente, si el nivel de juego es el complicado, aparecerán más estrellas que en el nivel de menos dificultad. La finalidad del juego consiste en destruir un super meteorito radiactivo que aparecerá en pantalla inmediatamente después de que hayan sido posicionadas las estrellas estáticas. Para ello, nuestro monitor radar está provisto de un indicador de posición a modo de cursor.

Pulsando cualquier tecla (excepto el 0), el indicador del monitor girará 45 grados en el sentido de las agujas del reloj, a medida que prosigue en su avance. Pulsando «0», dispararemos un rayo hiperlumínico capaz de destruir al más terco de los meteoros.

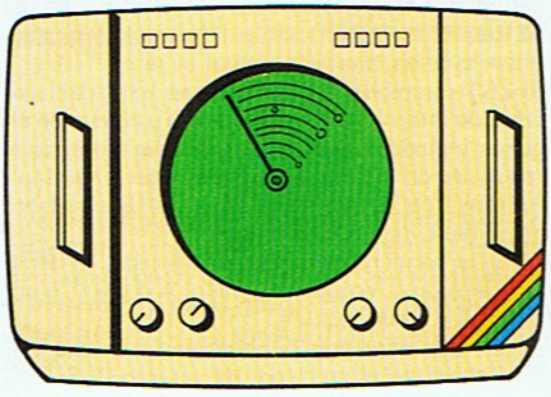
El indicador puede desplazarse en cualquier sentido del plano dependiendo siempre de su rotación de 45 grados. Si en nuestro avance, colisionamos con una estrella estática, el indicador variará de rumbo 45 grados, desviándose de la ruta de interceptación, aunque no se nos producirá daño alguno.

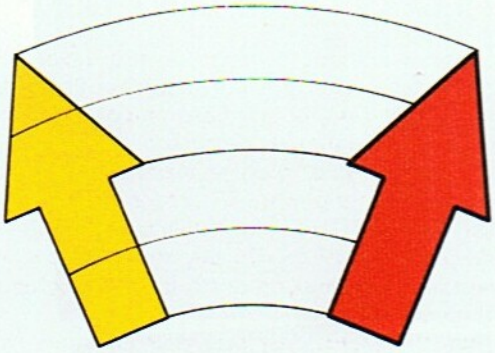
Para llevar a buen término la misión disponemos solamente de 100 segundos. Transcurrido el cual podemos considerarnos destruidos.

INTRODUCCION DEL PROGRAMA

La introducción del programa será incluso algo más sencilla de lo habitual, puesto que no exis-

Nuestro monitor o televisor acaba de convertirse en una pantalla de radar.





ten caracteres gráficos numéricos. Sólo tendremos que preocuparnos de sustituir las letras subrayadas en las líneas 1500 y 1700 del listado, por los caracteres gráficos correspondientes.
En lo referente a la grabación del programa, podemos llevarla a cabo mediante el comando **SAVE "RADAR"**, o bien **SAVE "RADAR" LINE 10** si deseamos que el programa se autoejecute al finalizar su carga.

Para el giro de 45 grados del cursor emplearemos cualquier tecla menos 0.

ALFONSO MENDEZ

```
 5 REM J.M. MAYORAL SERRANO
10 POKE 23658,8
20 GO SUB 790
30 GO SUB 1000
40 GO SUB 940
50 IF INT T>1000 THEN  GO TO 1800
60 GO SUB 110
70 GO SUB 540
80 IF H=1 THEN  GO TO 110
90 GO SUB 650
100 GO TO 40
110 LET A$=INKEY$
120 IF A$="" THEN  RETURN
130 IF A$="0" THEN  GO TO 170
140 LET K=K+1
150 IF K>8 THEN  LET K=1
160 RETURN
170 LET XL=X*8+4
180 LET YL=175-Y*8-4
190 GO SUB 470
200 PLOT XL,YL
210 LET DX=0
220 IF V(K)=1 THEN  LET DX=255-XL
230 IF V(K)=-1 THEN  LET DX=-XL
240 LET DY=0
250 IF W(K)=1 THEN  LET DY=-YL
260 IF W(K)=-1 THEN  LET DY=175-YL
270 IF V(K)*W(K)=0 THEN  GO TO 300
280  IF ABS DX<ABS DY THEN  LET DY=ABS DX*SGN DY: GO
TO 300
290 LET DX=ABS DY*SGN DX
300 DRAW DX,DY
310 PLOT XL,YL
320 BEEP .01,40
330 DRAW  OVER 1;DX,DY
340 IF H=0 THEN  RETURN
350 LET MX=B*8+4: LET MY=175-A*8-4
360 FOR I=1 TO RND*5+20
370 PLOT MX,MY
380 LET DX=10-RND*20
390 IF MX+DX>255 OR MX+DX<0 THEN  GO TO 450
400 LET DY=10-RND*20
410 IF MY+DY>175 OR MY+DY<0 THEN  GO TO 450
420 PLOT MX,MY
430 DRAW DX,DY
440 BEEP .01,10
450 NEXT I
460  GO TO 1810
470 LET H=0
480 LET DY=A-Y
490 LET DX=B-X
500 IF W(K)*DX<>V(K)*DY THEN  RETURN
510 IF ABS V(K)*SGN DX<>V(K) OR ABS W(K)*SGN DY<>W(K
) THEN  RETURN
520 LET H=1
530 RETURN
540 IF NB1=0 THEN  PRINT AT Y,X;" "
550 LET X=X+V(K)
560 LET Y=Y+W(K)
570 IF X<0 THEN  LET X=31
580 IF X>31 THEN  LET X=0
590 IF Y<0 THEN  LET Y=21
600 IF Y>21 THEN  LET Y=0
610 IF ATTR (Y,X)<>15 THEN  BEEP .1,0: LET NB1=1: GO
TO 140
620 PRINT AT Y,X;M$(K)
630 LET NB1=0
640 RETURN
650 IF RND>1.05-DF/20 THEN  LET Z=Z+1
660 IF Z>8 THEN  LET Z=1
670 IF NB2=0 THEN  PPINT AT A,B;" "
680 LET A=A+V(Z)
690 LET B=B+W(Z)
700 IF B<0 THEN  LET B=31
710 IF B>31 THEN  LET B=0
720 IF A<0 THEN  LET A=21
730 IF A>21 THEN  LET A=0
740  IF ATTR (A,B)<>15 THEN  BEEP .1,-10: LET NB2=1:
LET Z=Z+1: GO TO 660
750 PRINT AT A,B;W$(R+1)
760 LET R=NOT R
770 LET NB2=0
780 RETURN
790 PAPER 1
800 BORDER 1
810 INK 7
820 CLS
830 PPINT AT 15,1;"INTRODUCE NIVEL DE DIFICULTAD"
840  PRINT
850 PRINT TAB 3;"1 (Facil) ///  2 (Dificil)"
860  LET k$=INKEY$
870 IF k$="" THEN  GO TO 860
880  IF CODE k$<49 OR CODE k$>50 THEN  GO TO 860
890 BEEP .1,40
900 LET df=VAL k$
910  IF df=2 THEN  LET df=10
920 CLS
930 RETURN
940 LET T=(PEEK 23672+256*PEEK 23673+65536*PEEK 2367
4)/5
950 PRINT AT 0,0;INT T/10;"  "
960 RETURN
970 IF RND>.1+DF/50 THEN  RETURN
980 PRINT AT RND*21,RND*31; INK 6;"*"
990 RETURN
1000 DIM W(8)
1010 DIM V(8)
1020 DIM S$(8,8)
1030 DIM R$(8,8)
1040 DATA 0,1,1,1,1,0,1,-1,0,-1,-1,-1,-1,0,-1,1
1050 FOR I=1 TO 8
1060 READ W(I),V(I)
1070 NEXT I
1080 LET K=1
1090 LET X=20
1100 LET Y=10
1110 LET V=V(K)
1120 LET W=W(K)
1130 LET S$(1)="00011000"
1140 LET S$(2)="00111100"
1150 LET S$(3)="01111110"
1160 LET S$(4)="11111111"
1170 LET S$(5)="00111100"
1180 LET S$(6)="00111100"
1190 LET S$(7)="00111100"
1200 LET S$(8)="00111100"
1210 LET R$(1)="11111000"
1220 LET R$(2)="11110000"
1230 LET R$(3)="11111000"
1240 LET R$(4)="11111100"
1250 LET R$(5)="10111110"
1260 LET R$(6)="00011111"
1270 LET R$(7)="00001110"
1280 LET R$(8)="00000100"
1290 FOR I=1 TO 8
1300 LET A=0: LET B=0: LET C=0
1310 LET D=0: LET E=0: LET F=0
1320 FOR J=1 TO 8
1330 LET A=A*2+VAL (S$(I,J))
1340 LET B=B*2+VAL (S$(9-J,I))
1350 LET C=C*2+VAL (S$(J,I))
1360 LET D=D*2+VAL (R$(I,J))
1370 LET E=E*2+VAL (R$(9-J,I))
1380 LET F=F*2+VAL (R$(9-I,9-J))
1390 GO SUB 970
1400 NEXT J
1410 POKE USR "A"+I-1,A
1420 POKE USR "B"+8-I,A
1430 POKE USR "C"+I-1,B
1440 POKE USR "D"+I-1,C
1450 POKE USR "E"+I-1,D
1460 POKE USR "F"+8-I,D
1470 POKE USR "G"+I-1,E
1480 POKE USR "H"+I-1,F
1490 NEXT I
1500 LET M$="CHRFDEAG"
1510 LET A=10
1520 LET B=10
1530 LET Z=INT (RND*8)+1
1540  POKE USR "I"+0,BIN 00100000
1550 POKE USR "I"+1,BIN 01000010
1560 POKE USR "I"+2,BIN 00100101
1570 POKE USR "I"+3,BIN 00011000
1580 POKE USR "I"+4,BIN 00011000
1590 POKE USR "I"+5,BIN 10100100
1600 POKE USR "I"+6,BIN 01000010
1610 POKE USR "I"+7,BIN 00000100
1620 POKE USR "J"+0,BIN 00001110
1630 POKE USR "J"+1,BIN 10001000
1640 POKE USR "J"+2,BIN 10001000
1650 POKE USR "J"+3,BIN 11111000
1660 POKE USR "J"+4,BIN 00011111
1670 POKE USR "J"+5,BIN 00001001
1680 POKE USR "J"+6,BIN 00001001
1690 POKE USR "J"+7,BIN 00111000
1700 LET W$="IJ"
1710 LET R=0
1720 INK 7
1730 LET NB1=0
1740 LET NB2=0
1750 POKE 23674,0
1760 POKE 23673,0
1770 POKE 23672,0
1780 LET H=0
1790 RETURN
1800 IF H<>1 THEN  GO SUB 1840
1810 PRINT AT 13,9;" BUEN DISPARO "
1820 PRINT
1830 GO TO 1860
1840 PPINT AT 15,5; PAPER 6; INK 9;" EL TIEMPO HA ACA
BADO "
1850 PRINT
1860 PRINT TAB 5;" OTRA PARTIDITA? S/N"
1870 PRINT
1880 PRINT TAB 10;"TIEMPO= ";INT T/10
1890  LET K$=INKEY$
1900 IF K$="" THEN  BEEP .1,40: GO TO 1890
1910 IF K$="S" THEN  RUN
1920 IF K$="N" THEN  GO TO 10000
1930 BEEP .1,45
1940 GO TO 1890
```

¡!

Para la grabación del programa podemos utilizar el comando **SAVE "RADAR"**, o bien **SAVE "RADAR" LINE 10** si se desea la autoejecución del programa cada vez que se cargue.

*

Las letras que aparecen subrayadas en el listado corresponden a los gráficos de las teclas afectadas.

*

Aunque las teclas empleadas para el manejo del programa son todas las del teclado, este es enormemente sencillo de controlar.

ENSAMBLADORES

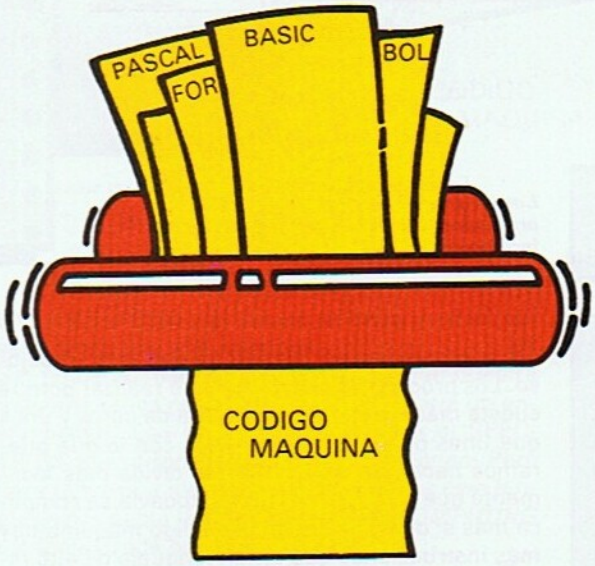
Si echamos un vistazo al mundo informático actual, nos daremos cuenta de la gran cantidad de lenguajes que nos rodean: empezando por diferentes dialectos del BASIC, uno por cada máquina (o incluso algunos ordenadores tienen posibilidad de incorporar diferentes versiones de BASIC por *software,* como es el caso del SPECTRUM), y terminando por las diferentes versiones de otros lenguajes con nombres de claves secretas de espías.

Si profundizamos un poco más en la inmensa «Torre de Babel» informática nos percataremos que cada lenguaje ha sido creado para un uso concreto: BASIC, para aprender; COBOL, para el manejo de gran cantidad de información; LOGO, para los niños; PASCAL, el superestructurado; FORTRAN, para los cálculos matemáticos... Sin embargo, por muchos lenguajes que nos rodeen y nos amenacen a todos, en el fondo, y para la máquina, se reducen a uno: el lenguaje máquina. Cuando estamos trabajando con un lenguaje de alto nivel, en el momento de ejecutarlo es necesario que el «micro» lo traduzca a código máquina, que es lo único que entiende nuestra Z-80. El proceso de traducción, sobre todo si el lenguaje que estamos utilizando es un intérprete, es el que retrasa todo programa no siendo lo suficientemente rápido para ciertas aplicaciones. Entonces ¿por qué no nos animamos a programar en código máquina?

Si ya nos hemos recuperado del susto por la anterior frase desafiante, intentaremos ver de una forma objetiva si es conveniente o no realizar un programa en código máquina.

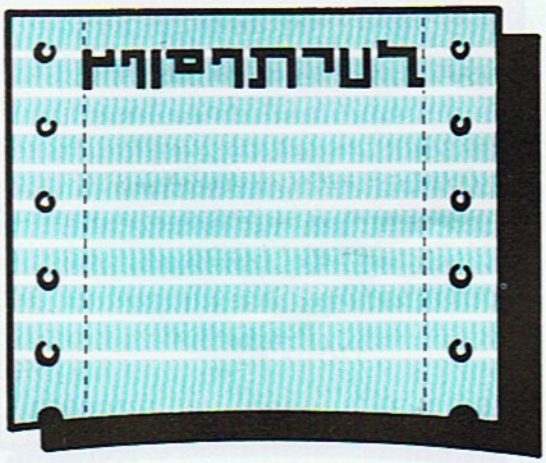
Entre las ventajas que nos aporta, la más llamativa es su velocidad. Un programa en este lenguaje es evidente que no necesita traducción, porque ya le hablamos a la máquina en el lenguaje que le gusta, y por tanto tarda bastante menos en ejecutar un programa (en algunos casos la velocidad es 60 veces superior a la del BASIC). Otra de las ventajas del lenguaje máquina es la poca memoria que ocupan los programas, y esto se nota sobre todo en los más largos.

Pero no todo iba a ser un camino de rosas, también hay inconvenientes. En caso de error de programación nos costará hallarlo, puesto que no tenemos mensajes de error, que siempre nos pueden ayudar a solucionar los problemas. Otra característica a apuntar es que aunque cada ordenador utilice BASIC's diferentes, con muy poco conocimiento de las particularidades de la máquina podemos adaptar programas para que funcionen en diferentes «micros»; sin embargo esto no es posible con el lenguaje máquina, porque nos obliga a conocer a fondo el funcionamiento



Para el ordenador todos los lenguajes son iguales, ya que se reducen al código máquina.

En la ejecución de un programa se pierde mucho tiempo en la traducción.

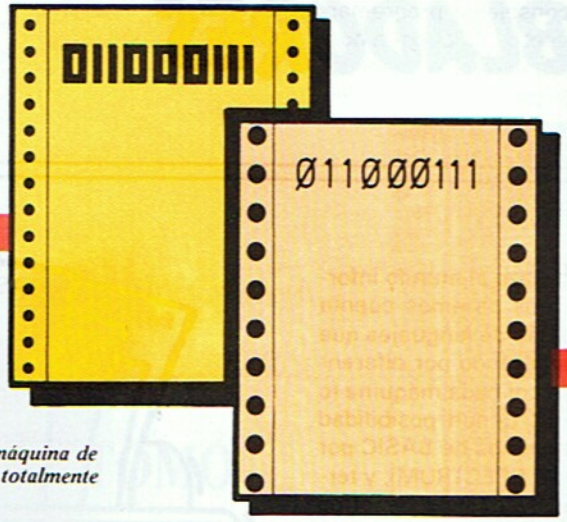


¡!

Cada lenguaje ha sido creado para un uso concreto, pero todos, en el fondo, se reducen a uno: el lenguaje máquina.

*

El proceso de traducción de un lenguaje de alto nivel es el que retrasa la ejecución de un programa, no siendo lo suficientemente rápido para ciertas aplicaciones.



Los programas en código máquina de ordenadores diferentes son totalmente incompatibles.

¡!

Los inconvenientes es la ausencia de los mensajes de error, la falta de compatibilidad entre diferentes «micros», dificultad de realización de los programas y mayor cantidad de instrucciones.

*

El ENSAMBLADOR sólo «sabe» realizar sumas y restas.

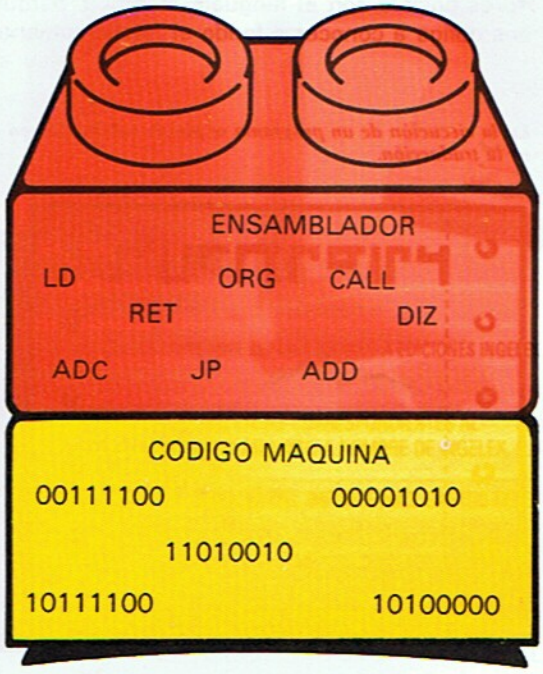
*

El ENSAMBLADOR es un lenguaje de ayuda a la programación en código máquina de bajo nivel porque cada mnemónico es equivalente a una instrucción en el código máquina.

del microprocesador siendo mucho más trabajoso. Los programas son difíciles de realizar porque cuesta más memorizar una ristra de ceros y unos que unas palabras relacionadas con lo que queremos hacer (así 00111100 se olvida más fácilmente que **LET A=A+1**), pero todavía se complica más si decimos que en el código máquina hay más instrucciones que en un lenguaje de alto nivel.

Sin embargo podemos casi solucionar el problema de programar con ceros y unos si tuviéramos comandos que nos recordasen la función que realizan. Esto es lo que nos aporta el lenguaje ENSAMBLADOR.

El ENSAMBLADOR es un lenguaje que tiene unos comandos que nos recuerdan qué función realizan y que se corresponden directamente con una instrucción del código máquina.



EL ENSAMBLADOR

Si a cada una de las 200 instrucciones del código máquina (formadas todas por ocho unos y ceros) las asociásemos un símbolo que nos recordarse la función que realizan y que fuera equivalente a cada comando del lenguaje máquina, nos costaría menos programar. Este conjunto de símbolos (que a partir de ahora llamaremos mnemónicos) es el que forma el lenguaje ENSAMBLADOR. Como último detalle, añadir que el mismo ENSAMBLADOR se encargará de realizar la traducción de los mnemónicos a los comandos del código máquina.

El ensamblador es un lenguaje de ayuda a la programación en código máquina de bajo nivel, porque cada comando, o mnemónico, es equivalente a una instrucción en el código máquina, y aunque sea necesaria una traducción, debido a la equivalencia que presenta apenas se nota una pérdida de velocidad.

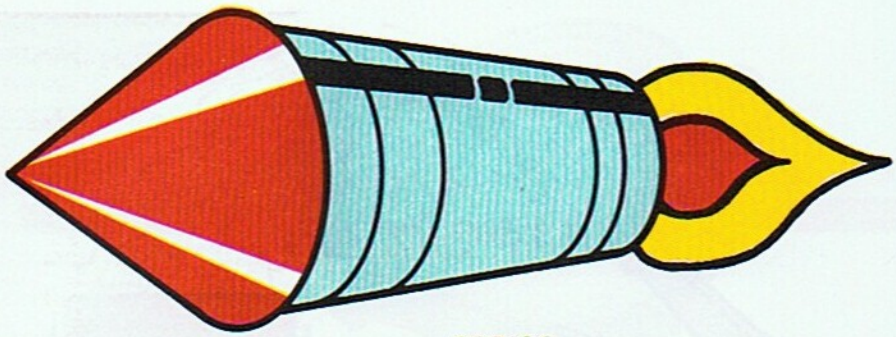
El lenguaje ensamblador aporta las mismas ventajas e inconvenientes que el código máquina, salvo la facilidad de recordar los mnemónicos, que aunque no son los comandos de los lenguajes de alto nivel, son mucho mejores que las series de números del código máquina.

Entonces, ¿será siempre aconsejable programar en ENSAMBLADOR? Es evidente que aunque nos facilita el trabajo no todo es tan sencillo como en el BASIC, por lo tanto hay que pensarse si en nuestro programa es conveniente utilizarlo o no. El ENSAMBLADOR (y también el código máquina) es bueno para todo aquello que necesite velocidad, por ejemplo para los juegos de *arcade* (los típicos de masacrar marcianitos), para manejar pantallas (cambios de paisaje en un juego), trabajar con ficheros, al ser posible encontrar un dato de forma muy rápida, para instrucciones que se repitan muy frecuentemente; al igual que para todo lo que necesite ahorro de memoria, como en los programas largos o ficheros. Por el contrario, no es nada aconsejable para todas aquellas cosas simples, como la impresión de mensajes por pantalla, puesto que en estos casos no se aprecia ninguna ventaja al hacerlo por BASIC o ENSAMBLADOR.

No existe una fórmula mágica para decidir en cual lenguaje programar, aquí hemos pretendido dar unas ideas, pero es el programador el que tiene la última palabra dependiendo del tiempo disponible para trabajar (el proceso es más largo, evidentemente, en el ensamblador), de la memoria que disponga nuestra máquina, la velocidad requerida, las necesidades que pretendamos satisfacer y otros factores que determinemos. Sin embargo, no es necesario realizar un programa entero en ENSAMBLADOR dado que podemos combinarlo perfectamente con el BASIC, y ahorrarnos el esfuerzo de «ensamblar» aspectos no interesantes.

PROFUNDIZAMOS

Ahora ya nos animamos más a programar en código máquina, porque nosotros lo hacemos en ENSAMBLADOR y él mismo se encargará en pasarlo traducido a la máquina.



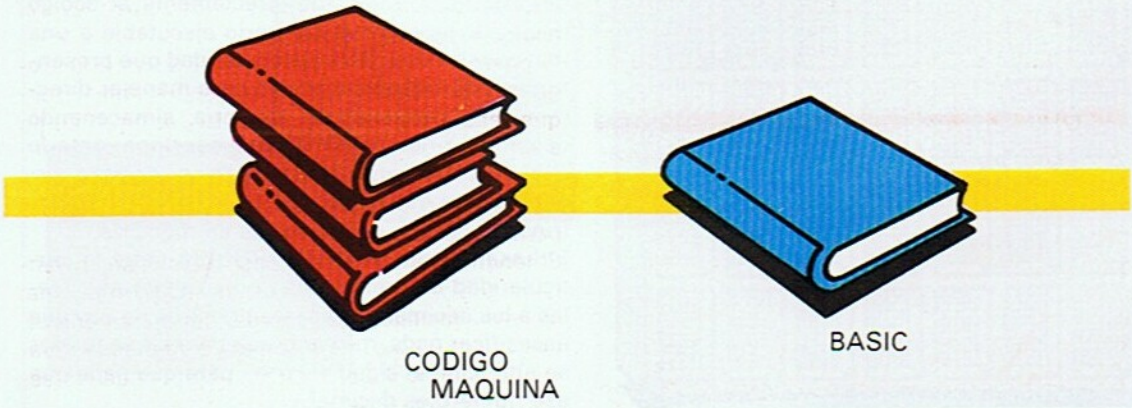
El lenguaje máquina es bastante más rápido que el BASIC.

Sin embargo, para ello necesitamos que nuestra máquina tenga posibilidad de incorporarle un ensamblador. Eso no es problema para el SPECTRUM que dispone de muchas versiones. Pero para tener una idea general de cómo es este lenguaje nos basta la versión de *ARTIC COMPUTING* comercializada por INDESCOMP.

Antes de continuar adelante es necesario diferenciar lo que es un «programa ensamblador» o un «programa en ensamblador». Lo primero se refiere al lenguaje en sí, es decir al programa encargado de traducir los mnemónicos a código máquina. El segundo es el programa que nosotros confeccionaremos, que será posteriormente «ensamblado por el ENSAMBLADOR».

También debemos diferenciar entre lo que es el código fuente y el código objeto. El primero es el programa que escribimos lenguaje ensamblador; el segundo es el mismo programa traducido a lenguaje máquina por el ensamblador.

En el código máquina tenemos más instrucciones que en un lenguaje de alto nivel.

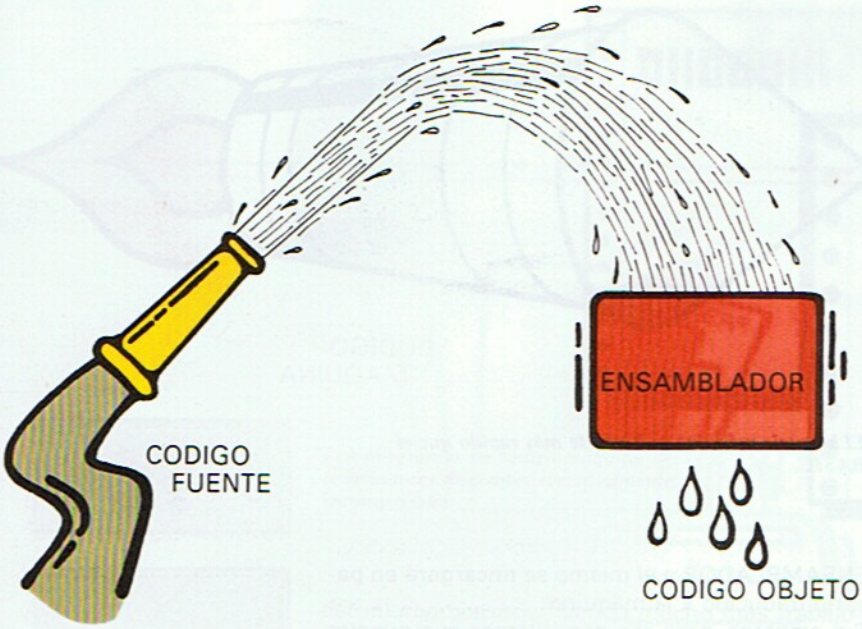


¡!

Las ventajas que nos ofrece el código máquina es la de su velocidad, hasta 60 veces más rápido que el BASIC, y el ahorro de memoria.

*

El ENSAMBLADOR presenta las mismas ventajas e inconvenientes que el código máquina, salvo la facilidad de recordar los mnemónicos.



El código fuente es el programa escrito por nosotros, y el código objeto es cuando está traducido a lenguaje máquina.

En este ensamblador el código fuente es situado automáticamente en una línea **REM** con el número 2. El código objeto lo hace también en una **REM** en la línea 1, a no ser que voluntariamente decidamos cambiar su emplazamiento con **ORG**. Por tanto, para grabar ambos códigos se hace como si fuera un programa BASIC y no como *bytes,* que es lo que hacen otros ensambladores.

Unos consejos importantes:

— En los programas largos grabar el código fuente y hacerlo por partes, para evitar disgustos cuando tengamos casi todo el programa tecleado (por ejemplo que nos quedemos sin luz).

— Antes de ejecutar un programa grabarlo. Si hay un error y se bloquea la máquina nos evitaremos teclear todo de nuevo.

— Cuando todo funcione ya se puede borrar el código fuente (**REM** línea 2 en el programa de **ARTIC**), aunque conviene guardar una copia de él para posibles rectificaciones posteriores.

Una de las diferencias entre el BASIC y el ENSAMBLADOR es que éste no utiliza variables. Sus «variables» son llamadas registros, y sirven para guardar la información provisionalmente. Es en ellas donde se realizan las operaciones aritméticas y lógicas. Por tanto, cada vez que se use la palabra registro debemos pensar en un espacio en la memoria para guardar la información.

Una vez cargado el ENSAMBLADOR estaremos directamente en el modo de inserción, pudiendo ya empezar a escribir nuestro programa. En caso de haber cometido algún error al escribirlo debemos entrar en el Editor para corregirlo. Para ello se pulsa **E** seguido de **ENTER** y tendremos acceso al menú que nos servirá para solucionar los problemas.

LOS PRIMEROS COMANDOS

Ya a estas alturas nos hemos podido percatar que el ensamblador es un lenguaje más, aunque tenga sus rarezas (todos los lenguajes las tienen). Los mnemónicos en ENSAMBLADOR realizan operaciones más simples, por eso es un lenguaje de bajo nivel, traducido directamente al código máquina, pero por el contrario ejecutable a una mayor velocidad. Otra particularidad que presentan es que tienen capacidad para manejar directamente direcciones de memoria, almacenando datos en donde consideremos oportuno, ante lo cual debemos tomar precauciones de dónde guardamos para que no se nos pierda nada por el intrincado bosque de las celdas de memoria.

El ensamblador de INVESTRONICA tiene la particularidad de preferir los números hexadecimales a los decimales. Para los primeros no hay que especificar nada, mientras que para los segundos se antepone el signo «+» o «–» para que sepa que ese número es decimal.

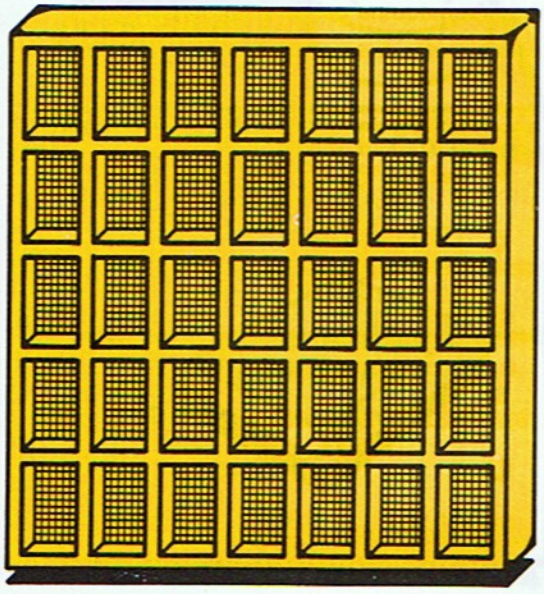
¡!

Un «programa ensamblador» es el encargado de traducir los mnemónicos a código máquina. Un «programa en ensamblador» es el que nosotros hacemos en este lenguaje.

*

Código fuente es el programa escrito por nosotros en ENSAMBLADOR. Código objeto es el mismo programa traducido (o ensamblado) a código máquina.

Registros es lo que usa el ENSAMBLADOR para guardar los datos.



Una de las instrucciones más similares a los lenguajes de alto nivel es **LD**, cuya función es la de asignación (el **LET** del BASIC). Su forma es:

LD A,2

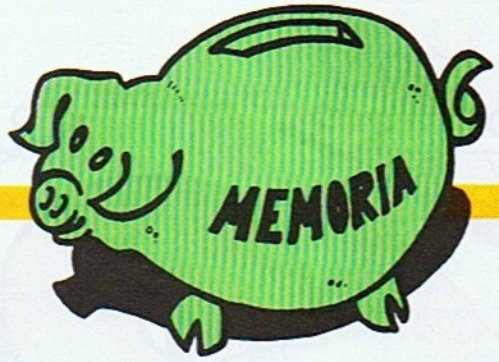
Aquí guardamos el valor «2» en el registro «A» (recordemos que no es lo mismo un registro que una variable, aunque, sin que se entere un informático, lo consideraremos equivalente).
Una variante del mismo comando es:

LD (HL),2

En esta ocasión guardamos el dato (recordemos que está en hexadecimal) dentro de la dirección de memoria indicada en el registro «HL». Es decir, en este registro tenemos almacenado un número el cual nos indicará el número de celda de memoria en que debemos guardar el dato que se encuentra a continuación. Creo que no ofrecerá ningún problema lo siguiente:

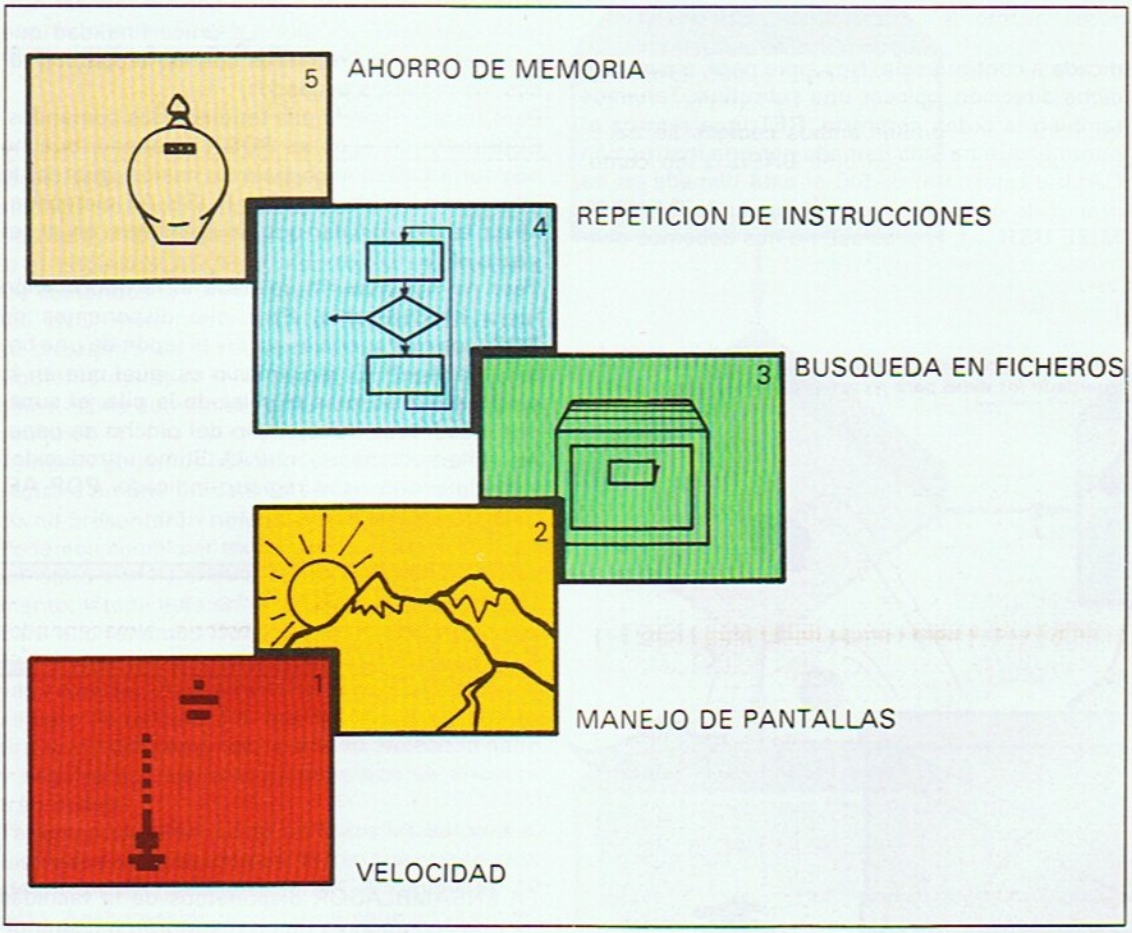
LD A,(HL)

¿Lo traducimos? Ahora guardaremos en el registro «A» el valor que se encuentra en la dirección de memoria indicada por «HL». Por tanto cada vez que utilicemos «()» se hará referencia a la dirección de memoria y no al valor almacenado.
La instrucción **CALL** supone un salto en la ejecución normal del programa, ya que éste se continuará ejecutando en la dirección de memoria in-



El lenguaje máquina presenta también la ventaja del ahorro de memoria.

Debemos analizar lo que vamos a programar para saber si resulta rentable hacerlo en ENSAMBLADOR.



¡!

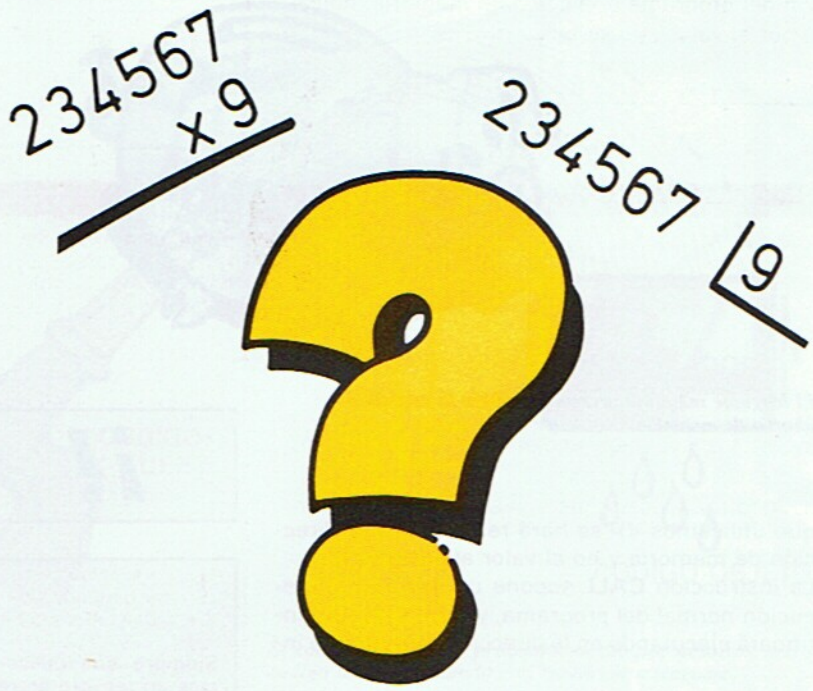
Siempre que utilicemos un registro entre «()» se refiere a la dirección de memoria indicada por el valor que tenga el registro dentro de «()».

*

Los mnemónicos realizan operaciones más simples pero más rápidas, y tienen la facultad de manejar directamente direcciones de memoria.

*

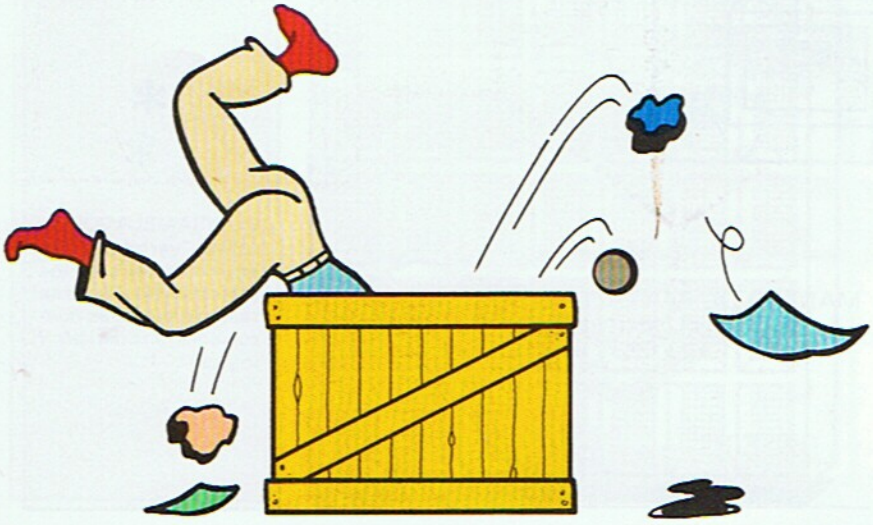
El ENSAMBLADOR no utiliza variables. Su equivalente son los registros, en donde se realizan todas las operaciones aritméticas y lógicas.



El ensamblador no «sabe» realizar otras operaciones diferentes a sumar y restar.

dicada a continuación. Nos sirve para, a partir de dicha dirección, colocar una subrutina. Tenemos también la orden contraria, **RET**, que retorna al punto al que ha sido llamado por una instrucción **CALL** o retorna al BASIC si está llamada se ha producido desde éste (con el comando **RANDOMIZE USR** ...), y, si es así, no nos debemos olvidar colocar este comando al final de nuestro programa en ensamblador para evitar que se nos quede «colgado» y no podamos recuperar el control del programa.

Debemos tener cuidado en qué celda de memoria hemos guardado los datos para recuperarlos cuando haga falta.



LA PILA

Es informática una pila es una forma de guardar provisionalmente unos datos numéricos. En el caso del ENSAMBLADOR se puede simular a esos archivadores que tienen un clavo largo donde se pinchan los papeles. Si clavamos varios papeles, el último que hayamos metido será el primero que podamos sacar (de una manera normal, se supone). A este tipo de pila se la conoce como *LIFO (Last In, First Out),* último dato en entrar, primero en salir.

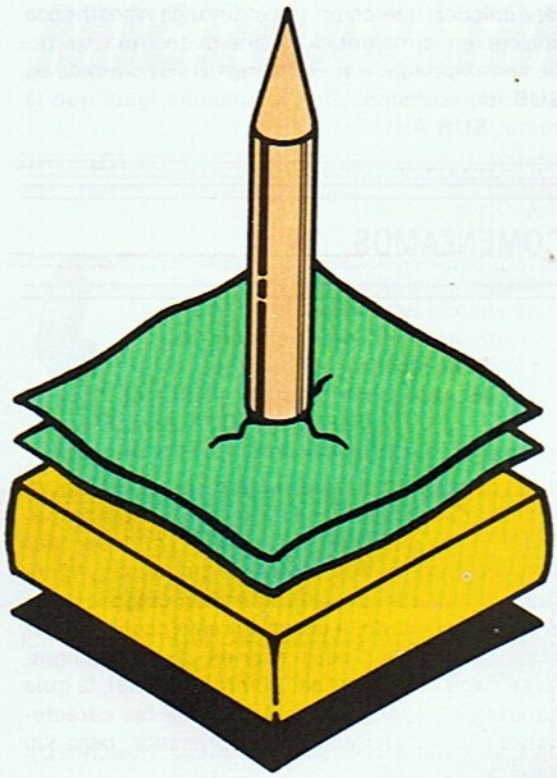
El ENSAMBLADOR utiliza la pila (también conocida como *stack*) para almacenar datos de una forma provisional no siendo posible realizar ninguna operación en ella. La única finalidad que tiene es guardar números cuando tengamos todos los registros ocupados.

Para trabajar con la pila tenemos dos comandos. El primero de ellos es **PUSH** («poner»), que no nos sorprenderá al conocer su misión: guardar la información en la pila. Así, **PUSH AF** almacenará en la pila el valor que se encuentre en el registro «AF».

Pero no nos serviría de nada si la información fuera irrecuperable. Para ello disponemos de **POP** (¿no nos suena a «sacar» el tapón de una botella de vino?). El mecanismo es igual que en la orden anterior: saca el valor de la pila, el superior (acuérdese del ejemplo del pincho de papeles, sólo podemos extraer el último introducido), y lo almacena en el registro indicado: **POP AF**.

INCREMENTAMOS Y DECREMENTAMOS

En ENSAMBLADOR disponemos de la facilidad de unos contadores que aumentarán o disminui-



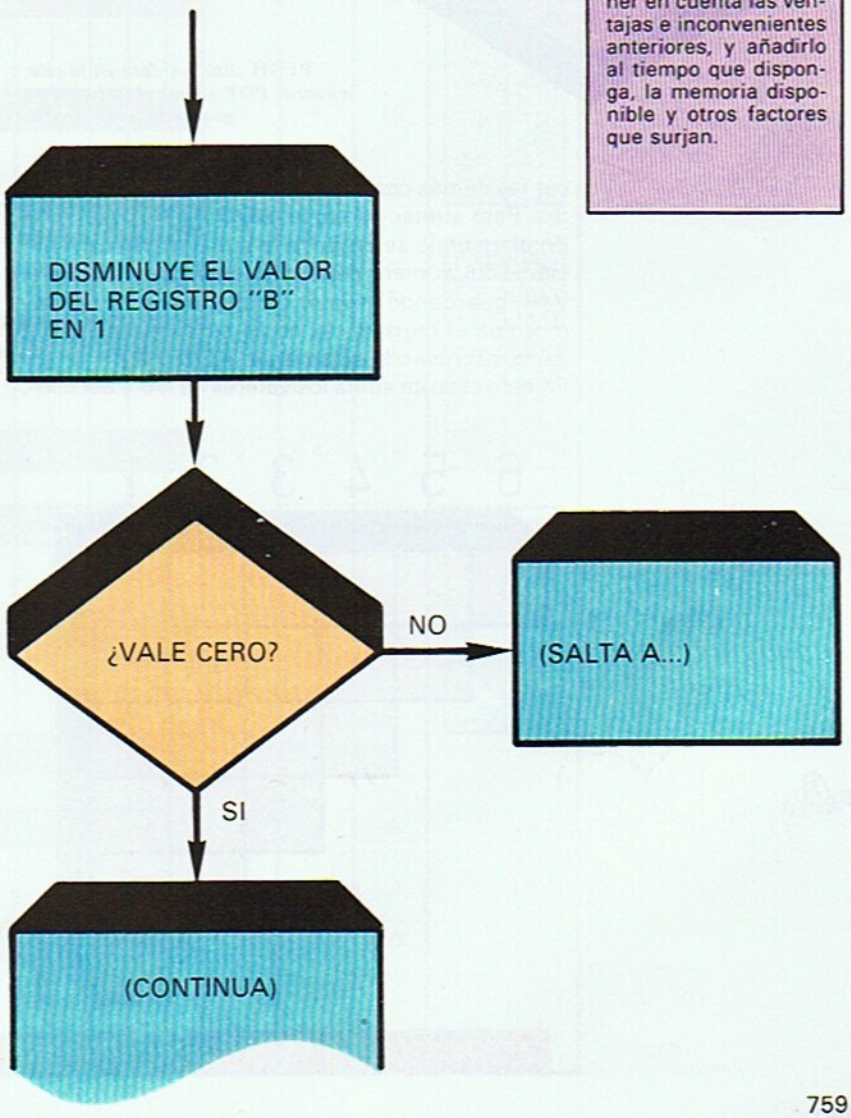
Una pila es un medio para guardar datos numéricos. El último dato en entrar en la pila es el primero en salir.

rán el valor de lo contenido en el registro al que se haga referencia en una unidad.
Aquí seguimos contando con la ventaja que aporta el ENSAMBLADOR al recordar la escritura de los comandos la función que realizan.
Así, **INC** «incrementa» en 1 el valor del dato que se encuentra en el registro indicado. Veamos el siguiente ejemplo: **INC HL.** El contenido del registro HL es aumentado en uno tras ejecutar este comando.
También podemos «decrementar» el valor de un registro en una unidad. La sintaxis y el significado no presentarán ningún problema: **DEC HL.**
Podemos complicar todavía más la situación con un salto condicional combinado con un decremento, y todo esto en un solo comando.
Para ello tenemos **DJNZ**, cuya función cada vez que se ejecuta es (¿preparados?) decrementar el valor que se encuentra en el registro «B» y si no es cero se ejecutará lo que se encuentre a continuación del comando, y si lo es no habrá salto y seguirá por las instrucciones que se encuentran debajo.
Tenemos también, como en todos los lenguajes, los saltos incondicionales. El representante más simple es **JP** (del inglés *jump*). Por ejemplo, **JP (HL).** Acabamos de producir un salto en la ejecución del programa a la celda de memoria indicada por el valor que se encuentra en «HL».

SUMAMOS Y RESTAMOS

No nos llevaremos ninguna sorpresa si le decimos que con el ensamblador podemos sumar y restar. Aunque sí posiblemente si le decimos que sólo «sabe» realizar estas dos operaciones, y en realidad no necesita más, porque se pueden ha-

Aquí se ve la función de DJNZ.

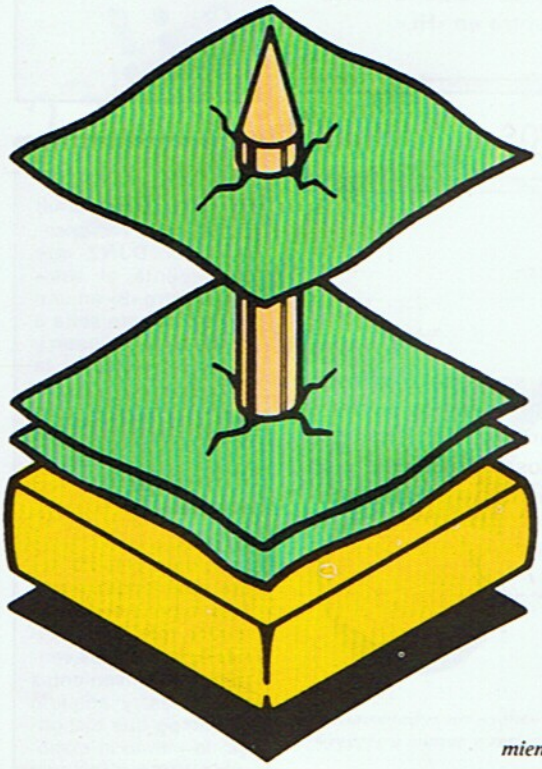


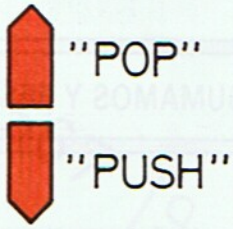
¡!

El salto condicional puede estar representado por **DJNZ** que decrementa el valor del registro «B» en uno y si no es cero salta a donde se le indiquen y si lo es continua la ejecución del programa.

*

La última palabra sobre cual lenguaje utilizar la tiene el programador, que deberá tener en cuenta las ventajas e inconvenientes anteriores, y añadirlo al tiempo que disponga, la memoria disponible y otros factores que surjan.





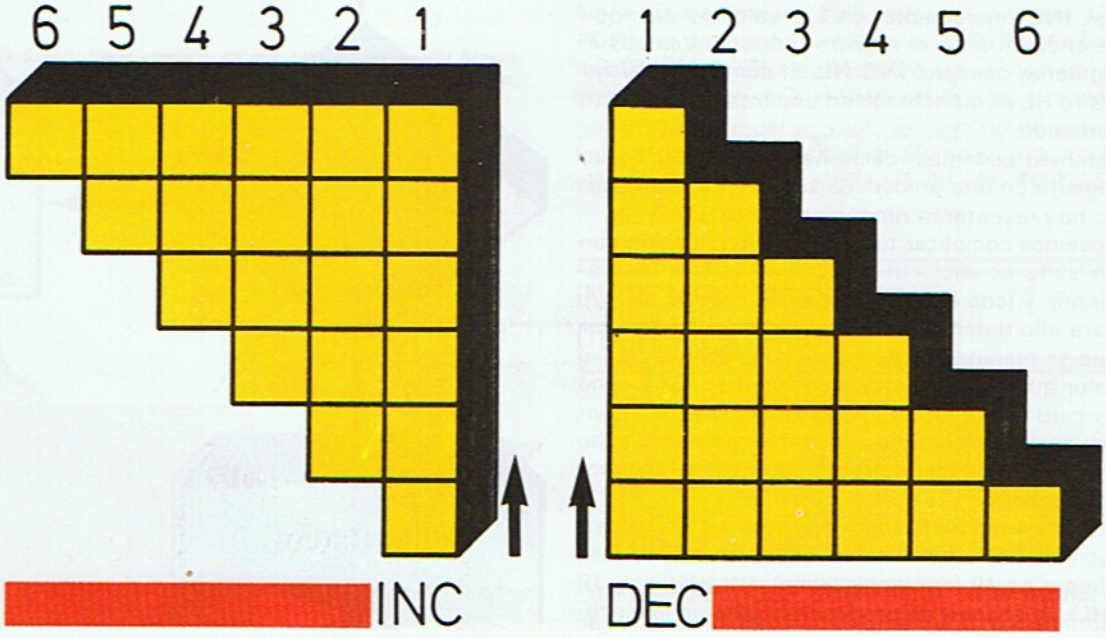
PUSH *sitúa un dato en la pila, mientras* **POP** *extrae el último dato guardado, el de arriba.*

cer las demás como combinación de sumas y restas. Para sumar haremos: **ADD A,H.**
En el ejemplo se produce la suma (**ADD** de «adición») del número contenido en los registros «A» y «H» guardando el valor en el primero. Si operamos con el registro «A» no es necesario especificarlo, escribiendo solamente: **ADD (HL).**
En este caso se suma los valores de «A» y del contenido en la dirección de memoria especificada por «HL», guardando el resultado en «A».
De forma parecida opera la resta; su comando es **SUB** (de «substracción», y funciona igual que la suma: **SUB A,H.**

COMENZAMOS

Nuestro objetivo al tratar el tema de este lenguaje es quitar el miedo que existe con el código máquina y con el ENSAMBLADOR, e intentar presentarlo como lo que es: un lenguaje más, necesitando para ello un programa ensamblador como el aquí comentado, a pesar, en este caso, que por culpa de las «mini-instrucciones» que le acompaña se haga cuesta arriba. Como es normal, la guía no enseña el lenguaje, sólo explica las características y peculiaridades que presenta, pero sin muchos detalles.
Esperemos que esa caja negra que es para muchos el ENSAMBLADOR, haya desaparecido, y se anime a hacer sus primeros pinitos, no conformándose con lo aquí expuesto, sino aprovechar este final para comenzar a «ensamblar».

INC *aumenta en uno el valor de lo contenido en el registro y* **DEC** *disminuye dicho valor en uno.*



MICROARITMETICA

La ALU (unidad aritmética y lógica) situada en el interior del Z 80 es la encargada de gestionar, entre otros trabajos, aquellos que impliquen la ejecución de sumas, restas y manejo de los operadores lógicos AND, OR y XOR.
Estos últimos ya fueron discutidos en el capítulo anterior, por lo que en este nos centraremos en las instrucciones aritméticas previstas por los fabricantes del microprocesador instalado en nuestro Spectrum.

Grupo aritmético de 16 bits.

MNEMONICO	CODIGO MAQUINA	REGISTRO F 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Nº BYTES	CICLOS MAQ.	CICLOS RELOJ	NOTAS
ADD HL, ss	0 0 s s 1 0 0 1	●	●		x		●	0	↕	1	3	11	
ADC HL, ss	1 1 1 0 1 1 0 1 0 1 s s 1 0 1 0	↕	↕		x		v	0	↕	2	4	15	
SBC HL, ss	1 1 1 0 1 1 0 1 0 1 s s 0 0 1 0	↕	↕		x		v	1	↕	2	4	15	
ADD IX, pp	1 1 0 1 1 1 0 1 0 0 p p 1 0 0 1	●	●		x		●	0	↕	2	4	15	
ADD IY, rr	1 1 1 1 1 1 0 1 0 0 r r 1 0 0 1	●	●		x		●	0	↕	2	4	15	
INC ss	0 0 s s 0 0 1 1	●	●		●		●	●	●	1	1	6	
INC IX	1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 1	●	●		●		●	●	●	2	2	10	
INC IY	1 1 1 1 1 1 0 1 0 0 1 0 0 0 1 1	●	●		●		●	●	●	2	2	10	
DEC ss	0 0 s s 1 0 1 1	●	●		●		●	●	●	1	1	6	
DEC IX	1 1 0 1 1 1 0 1 0 0 1 0 1 0 1 1	●	●		●		●	●	●	2	2	10	
DEC IY	1 1 1 1 1 1 0 1 0 0 1 0 1 0 1 1	●	●		●		●	●	●	2	2	10	

Reg.	ss
BC	00
DE	01
HL	10
SP	11

Reg.	PP
BC	00
DE	01
IX	10
SP	11

Reg.	rr
BC	00
DE	01
IY	10
SP	11

GRUPO ARITMETICO DE 8 BITS

Pertenecen a este grupo, todas aquellas instrucciones que efectúen la suma (ADD) o resta (SUB) de un valor contenido en un registro o posición de memoria con el almacenado en el acumulador, quedando el resultado en este último y manteniéndose inalterado el valor del registro o posición de memoria implicada tras la operación aritmética.

Genéricamente podríamos representar estas operaciones de la manera siguiente:

SUMA A+n->A
RESTA A-n->A

MNEMONICO	CODIGO MAQUINA	REGISTRO F 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Nº BYTES	CICLOS MAQ.	CICLOS RELOJ	NOTAS
ADD r	1 0 0 0 0 r r r	↕	↕		↕		v	0	↕	1	1	4	
ADD n	1 1 0 0 0 1 1 0 n n n n n n n n	↕	↕		↕		v	0	↕	2	2	7	
ADD (HL)	1 0 0 0 0 1 1 0	↕	↕		↕		v	0	↕	1	1	4	
ADD (IX+d)	1 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 d d d d d d d d	↕	↕		↕		v	0	↕	3	5	19	
ADD (IY+d)	1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 0 d d d d d d d d	↕	↕		↕		v	0	↕	3	5	19	
ADC s	0 0 1	↕	↕		↕		v	0	↕	(a)	(b)	(c)	
SUB s	0 1 0	↕	↕		↕		v	1	↕				
SBC s	0 1 1	↕	↕		↕		v	1	↕				
INC r	0 0 r r r 1 0 0	↕	↕		↕		v	0	●	1	1	4	
INC (HL)	0 0 1 1 0 1 0 0	↕	↕		↕		v	0	●	1	3	11	
INC (IX+d)	1 1 0 1 1 1 0 1 0 0 1 1 0 1 0 0 d d d d d d d d	↕	↕		↕		v	0	●	3	6	23	
DEC s	1 1 1 1 1 1 0 1 0 0 1 1 0 1 0 0 d d d d d d d d	↕	↕		↕		v	1	●	3	6	23	

Reg.	
A	111
B	000
C	001
D	010
E	011
H	100
L	101

S
r
n*
(HL)
(IX+d)
(IY+d)

* Salvo en DECs

- ● Indicador no afectado. Indicador modificado según el resultado de la operación.
- P El bit 2 del registro F actúa como indicador de pondad (1 – pondad par/0 – pondad impar).
- V El bit 2 del registro F actúa como indicador de sobrepasamiento (1 – sobrepasam./ 0 – no sobrepasam.).
- X Estado indeterminado.

(a) (b) (c) En las instrucciones donde se utiliza la abreviatura "s" el nº de bytes, ciclos máquina y de reloj, son los mismos que sus correspondientes anteriores.

Grupo aritmético de 8 bits.

donde A representa el valor almacenado en el acumulador, y n el contenido del registro o posición de memoria especificado en la instrucción. Además, es posible incrementar (INC) o decrementar (DEC) en 1 la cantidad almacenada en un determinado registro. Veamos el efecto que sobre los indicadores tienen estas operaciones:

S.—Cuando el resultado en A es negativo, es decir, si el bit más significativo de éste es 1, la bandera de signo contendrá un 1. En caso contrario será 0.

Z.—Tras una operación aritmética, su valor será 1, si el resultado de ésta fue 0. De lo contrario, contendrá un 0.

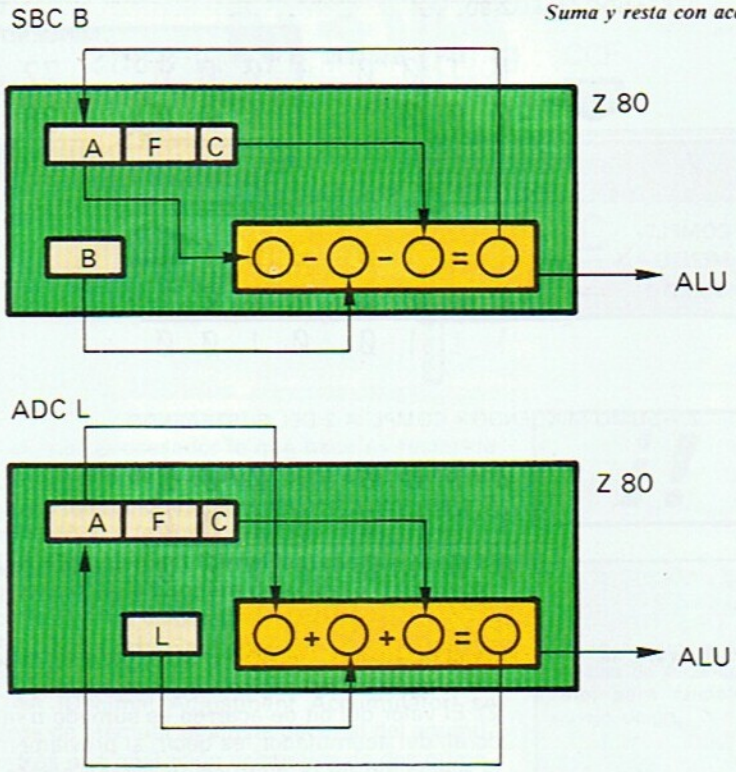
N.—El indicador de resta se coloca a 1 en todas las operaciones relacionadas con esta operación, mientras que en las sumas permanece a 0.

P/V.—Señala si hubo sobrepasamiento (1) o no (0) en las operaciones aritméticas de dos números expresados en complemento a dos.

C.—Actúa de manera similar al anterior, pero con números expresados en su correspondiente formato binario.

H.—Su importancia se pone de manifiesto cuando ejecutamos operaciones aritméticas con números en formato BCD (decimal codificado en binario), los cuales serán discutidos próximamente, al analizar la instrucción DAA.

En el capítulo dedicado a la presentación de las tablas de instrucciones, se señalan los operandos sobre los cuales actúan las de mnemónico ADD, SUB, INC y DEC. Como podemos comprobar, además están contempladas dos operaciones especiales para la suma y la resta, las cuales tienen presente el valor de la bandera de arrastre C. Se trata de aquellas cuyo mnemónico es ADC (sumar con acarreo) y SBC (restar con acarreo).



Suma y resta con acarreo.

Implementar estas instrucciones obliga al microprocesador a seguir los siguientes pasos basados en el esquema:

SUMA con acarreo A+n+C->A

RESTA con acarreo A-n-C->A

1) Suma o resta al/del acumulador el byte especificado en la instrucción como si se tratara, al

MNEMONIACO	CODIGO MAQUINA	REGISTRO F 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Nº BYTES	CICLOS MAQ.	CICLOS RELOJ	NOTAS
DAA	0 0 1 0 0 1 1 1	↕	↕		↕		P	●	↕	1	1	4	
CPL	0 0 1 0 1 1 1 1	●	●		1		●	1	●	1	1	4	
NEG	1 1 1 0 1 1 0 1 0 1 0 0 0 1 0 0	↕	↕		↕		v	1	↕	2	2	8	
CCF	0 0 1 1 1 1 1 1	●	●		x		●	0	↕	1	1	4	
SCF	0 0 1 1 0 1 1 1	●	●		0		●	0	1	1	1	4	
NOP	0 0 0 0 0 0 0 0	●	●		●		●	●	●	1	1	4	
HALT	0 1 1 1 0 1 1 0	●	●		●		●	●	●	1	1	4	

Grupo aritmético de la C.P.U.

RESTANDO EN EL Z-80

```
                                          MINUENDO
  Ø 1 Ø Ø  1 Ø Ø Ø   72
 -Ø Ø Ø 1  1 1 Ø Ø  -28
 ----------------------
                     SUSTRAENDO
```

1.—COMPLEMENTO A 2 DEL SUSTRAENDO

```
1.1. COMPLE-
     MENTO A 1 →  1 1 1 Ø  Ø Ø 1 1
1.2. SUMO 1   →  +Ø Ø Ø Ø  Ø Ø Ø 1
                 ----------------
                  1 1 1 Ø  Ø 1 Ø Ø
```

2.—SUMO MINUENDO + COMPL. A 2 DEL SUSTRAENDO

```
  Ø 1 Ø Ø  1 Ø Ø Ø
 +1 1 1 Ø  Ø 1 Ø Ø
 ----------------
  Ø Ø 1 Ø  1 1 Ø Ø  → 44 DECIMAL
```

Restando en el Z 80.

igual que antes, de ADD o SUB, pero sin modificar ningún bit del registro de indicadores.

2) El valor del bit de acarreo es sumado o restado al/del acumulador, es decir, si previamente a la ejecución de la instrucción estaba alzado (1), se suma o resta 1 de A.

3) Finalmente el Z 80 ajusta las banderas del registro F, basándose en el último resultado recogido en A.

El BCD (Binary Coded Decimal) es un sistema para codificar números decimales empleando 4 dígitos binarios por cada uno decimal.

Cuando trabajamos en aritmética BCD, tras cualquier operación debemos ejecutar una instrucción DAA para evitar posibles confusiones.

EL GRUPO ARITMETICO DE 16 BITS

Las instrucciones aritméticas anteriormente comentadas ADD, ADC, SBC, INC y DEC, que actúan sobre registros o posiciones de memoria de 8 bits, tienen sus análogas cuando se trata de operar con 16 bits, pero solamente trabajan con cantidades almacenadas en registros dobles, no siendo posible recogerlas de una celda de memoria determinada.

En el caso de las sumas y restas con 8 bits, el primer operando no era necesario especificarlo en la instrucción, pues el microprocesador tomaba siempre por defecto el valor almacenado en el acumulador.

Cuando se trata de sumar o restar números de 2 bytes, el primer operando ha de estar contenido en el par HL, salvo en las instrucciones ADD IX,pp y ADD IY,rr donde se toman los registros índice con este fin.

El segundo operando será el valor almacenado en cualquier par de registros de los autorizados en la instrucción en concreto de que se trate (ss, pp o rr), siguiéndose la decodificación expuesta en la tabla, sin que sea posible trabajar con parejas como BL o HC.

Tras efectuar la suma o resta, el resultado queda almacenado en HL (en IX o IY si es su caso), manteniéndose inalterado el contenido del par de registros utilizado como segundo operando.

Aclaremos su funcionamiento con algunos ejemplos: supongamos que deseamos ejecutar la instrucción ADD HL,DE siendo el valor contenido en HL 52277 y en DE 23758:

	ANTES	DESPUES
H	11001100	00101001
L	00110101	00000011
D	01011100	01011100
E	11001110	11001110

Es decir, tras sumar 52277 y 23758 ¡hemos obtenido como resultado 10499! en vez de 76035 como sería de esperar. Podríamos pensar que el Z 80 no tiene ni idea de lo que es una suma. Si nos fijamos en la tabla donde se detallan los indicadores afectados observaremos que sólo tres de ellos pueden haber cambiado.

La bandera de semiacarreo (H) se alzará en estas operaciones, cuando al ajecutar la suma binaria nos vayamos «llevando una» desde el bit 11 hasta el final. En nuestro ejemplo no ocurre esto y por tanto, permanecerá a 0. El indicador de resta N contendrá 1 cuando efectuemos, por ejemplo, una instrucción SBC, es decir, una resta. En nuestro caso la operación implementada fue una suma y contendrá 0, por tanto.

Hasta ahora no hemos conseguido detectar ninguna condición que permita justificar tan sorprendente resultado y sólo nos queda uno de los bits del registro F: el representativo del acarreo. Esta bandera se alza cuando se produce acarreo en el bit 15 (si me «llevo una» a partir del bit 15 al realizar la suma). Y eso precisamente es lo que ha ocurrido en nuestro ejemplo (comprobémoslo). Entonces el bit de acarreo C del registro F contendrá 1.

¿Cómo interpretamos entonces el resultado? Todos sabemos que con 16 bits podemos representar números decimales entre 0 y 65535, es decir 65536 combinaciones distintas. Pues 10499 es exactamente la cantidad resultante de efectuar la operación 52277+23758-65536.

Como el par HL sólo puede contener números de 16 bits, es decir, inferiores a 65535, el 76035 se le hace demasiado grande y por ello le resta 65536, circunstancia que detectamos al comprobar que el banderín de acarreo está alzado.

Las instrucciones de incremento y decremento de 16 bits son completamente análogas en su funcionamiento a sus correspondientes de 8 bits, sólo que ahora actúan sobre registros dobles.

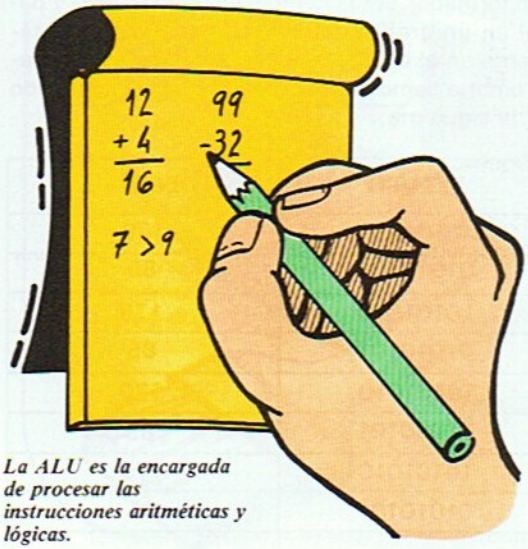
Al ejecutarlas se suma/resta 1 del contenido del byte de menor peso dentro del par especificado en la instrucción. Si esta operación provoca acarreo, éste es transmitido automáticamente al octeto superior.
En los programas en C/M se utilizan para realizar bucles de más de 256 repeticiones, donde puede resultar interesante conocer cuando hemos llegado, a base de decrementar un valor prefijado, a 0.
Como estas instrucciones no afectan al contenido del registro de indicadores podríamos pensar que la anterior circunstancia no es detectable. Sin embargo, podemos servirnos del siguiente artificio: supongamos que hemos estado decrementando el par BC. Tras ejecutar las instrucciones

```
LD A,B
OR C
```

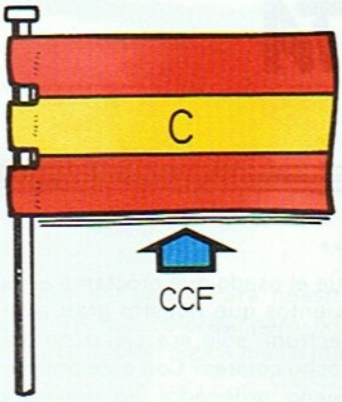
la bandera de cero **Z** contendrá un 1, solamente cuando tanto B como C sean 0.

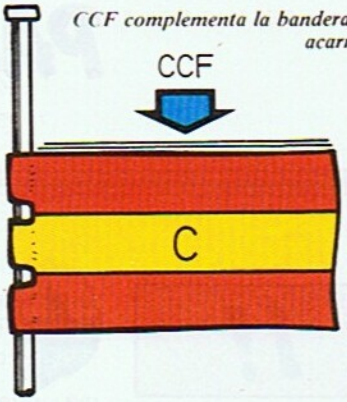
EL GRUPO ARITMETICO DE LA CPU

Pertenecen a éste algunas de las instrucciones incluidas en el capítulo 41 bajo la denominación «aritmético y de control de la CPU», las cuales vienen a complementar a las anteriores. Comentemos cual es exactamente su cometido:
— CPL (ComPLement, complementar) efectúa el complemento a 1 del valor almacenado en el acumulador, es decir, cambia los 1 por 0 y los 0 por 1.
— NEG (NEGative, negativo) realiza la negación del acumulador, operación que ha de entenderse como encontrar su complemento a 2. En realidad, el microprocesador lo que hace es restar de 0 el contenido de A (0-A), y para ello, como en todas las operaciones de resta complementa a 2 el sustraendo y le suma después el minuendo.



La ALU es la encargada de procesar las instrucciones aritméticas y lógicas.





CCF complementa la bandera de acarreo.

— CCF (Complement Carry Flag) complementa el valor actual de la bandera de acarreo C, es decir, si contenía 1, la coloca 0 y viceversa.
— SCF (Set Carry Flag) pone a 1 el indicador de acarreo C.
— DAA (Decimal Adjustment Accumulator) se encarga de efectuar el ajuste decimal del acumulador tras una operación aritmética de dos números en formato BCD.
El BCD es un sistema de codificación encargado de representar cada dígito correspondiente a un número decimal mediante cuatro dígitos binarios. Por ejemplo, el número decimal 76 tendría el siguiente aspecto en BCD:

01110110

Si quisiéramos representar un número mayor, por ejemplo, 6592 necesitaríamos dos octetos: 01000101 y 10010010.
Al realizar una operación aritmética con números en este formato, lo normal es que el resultado recogido en el acumulador no esté en BCD y para transformarlo a éste, se utiliza la instrucción DAA, de manera que no se produzcan errores de interpretación entre cantidades en binario y BCD.
A la anteriores podemos añadir dos instrucciones, las cuales afectan directamente a la CPU. Se trata de NOP (No OPeration) la cual simplemente indica al Z 80 que no haga nada. Esto no quiere decir que ejecutar esta instrucción no tome tiempo, pues de hecho se requieren 4 ciclos de reloj para llevarla a cabo. Su utilidad versa en proporcionar espacio dentro de un programa al objeto de añadir luego otras instrucciones.
Finalmente la orden HALT provoca que el microprocesador pare cualquier trabajo que esté efectuando.
Con esto cerramos el análisis de los grupos aritmético y lógico del Z 80 emprendido en el capítulo anterior. En el próximo discutiremos las instrucciones que provocan bifurcaciones dentro de la secuencia de un programa.

¡!

La instrucción NOP precisa de 4 ciclos de reloj para completar su ejecución.

*

Las operaciones aritméticas entre números de 16 bits, sólo es posible efectuarlas cuando estos se encuentran almacenados en registros dobles.

*

El resultado de una operación aritmética de 8 bits siempre queda almacenado en el acumulador.

*

HALT provoca la detención en cualquier trabajo que el microprocesador esté efectuando.

PALETA

¡!

Para la grabación del programa emplearemos el comando **SAVE "PALETA"**, o bien, **SAVE "PALETA" LINE 10**, si deseamos que éste se ejecute automáticamente al finalizar su carga.

*

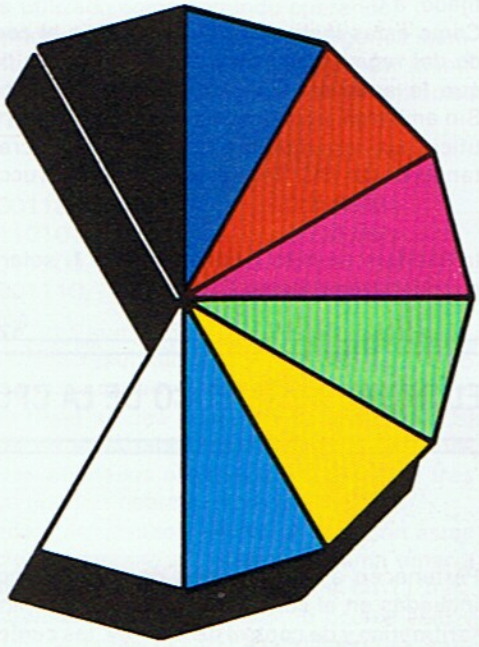
Los caracteres **A** subrayados en las líneas 1740 y 1920 del listado, corresponden al gráfico definido de dicha tecla.

Quién fue el osado que proclamó a los siete vientos que nuestro gran amigo Spectrum, sólo era capaz de generar ocho colores? Con este programa, nuestro pequeño ordenador no generará ocho, ni cuarenta, ni ochenta colores, sino ciento veintisiete. ¿Asombrados?

EL PROGRAMA

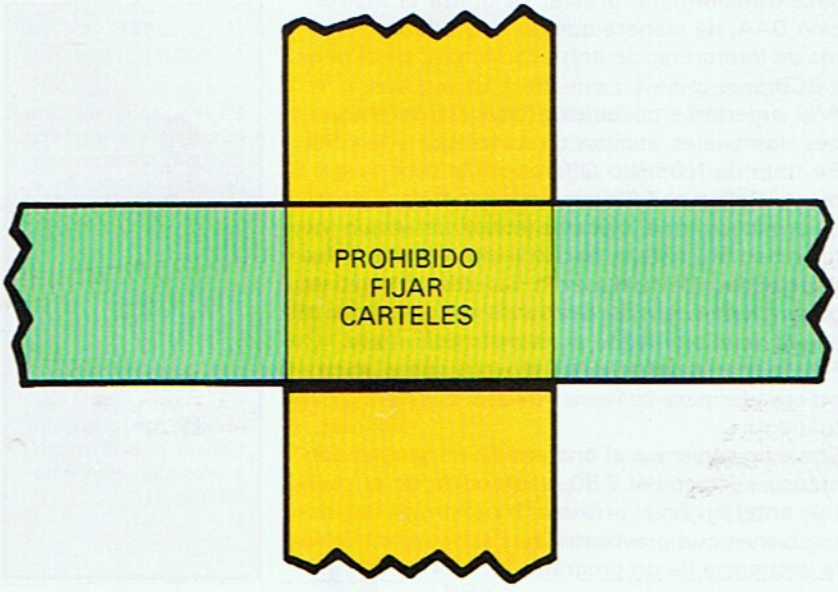
Nuestro programa PALETA genera ciento diecinueve mezclas, partiendo de los ocho colores básicos con los que normalmente puede trabajar el Spectrum. Estos ciento diecinueve tonos, junto con los ocho anteriormente mencionados forman el conjunto de ciento veintisiete colores.

Tanto los tonos primarios como las mezclas, están formados por la combinación de la tinta y papel en un gráfico definido de usuario, concretamente en el U.D.G. de «A». La configuración tanto binaria como decimal de dicho gráfico definido es la siguiente:



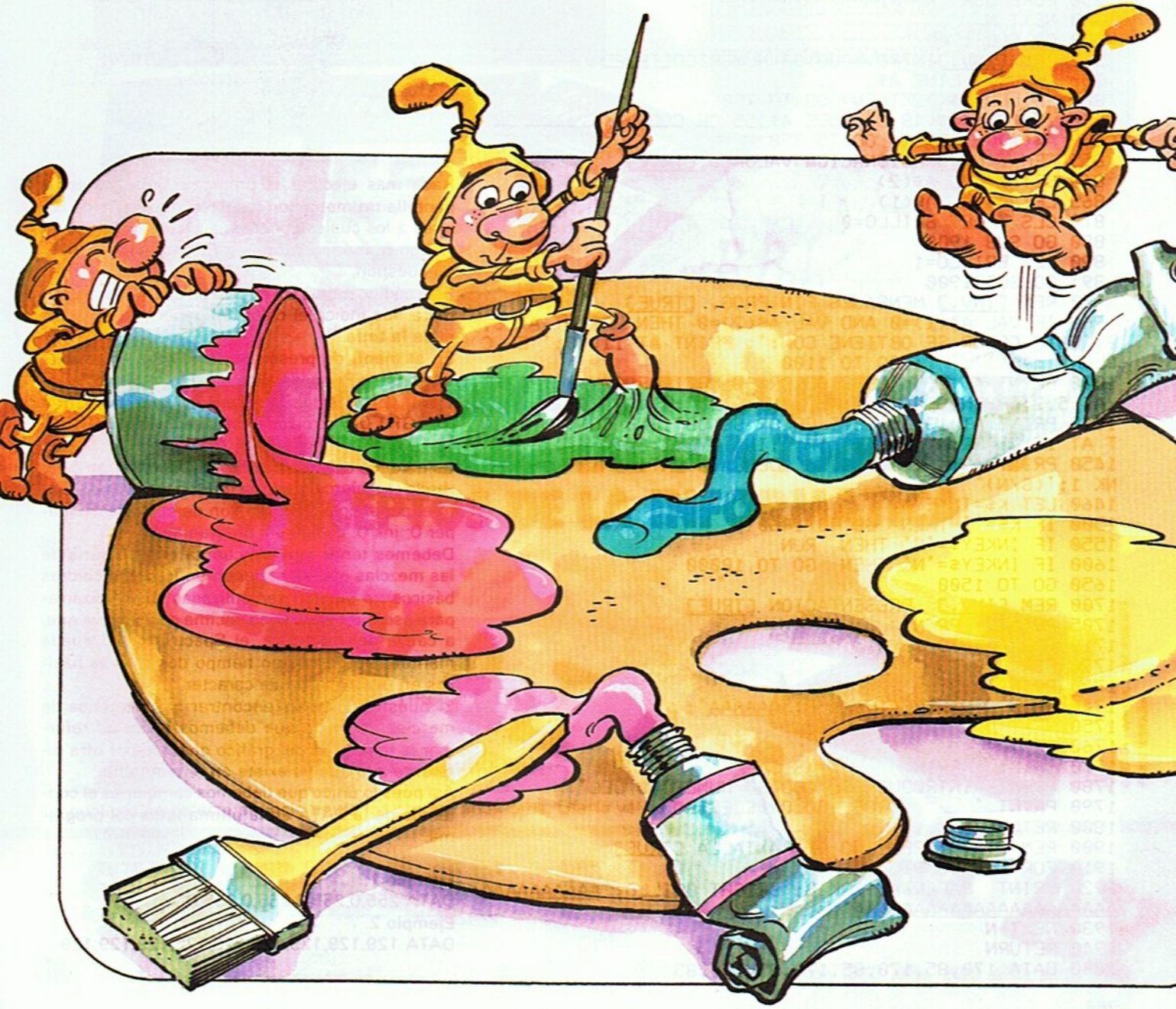
Nuestro programa PALETA genera 119 mezclas, partiendo de los 8 colores básicos del Spectrum.

Al ejecutar el programa aparece en la pantalla un menú con los diferentes tipos de colores, a los cuales podremos acceder mediante un código numérico.



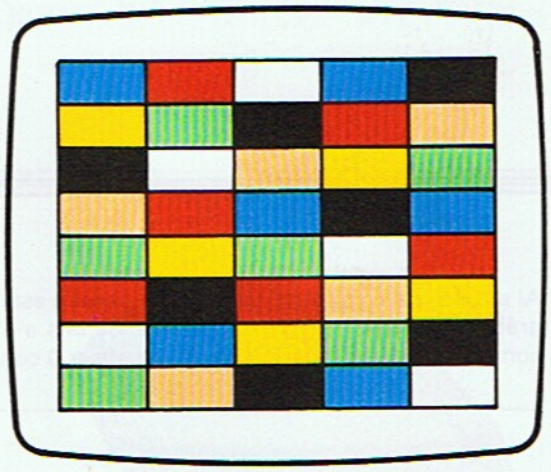
BINARIO	DECIMAL
10101010	170
01010101	85
10101010	170
01010101	85
10101010	170
01010101	85
10101010	170
01010101	85

Al asignar un color de tinta y otro de papel a este gráfico definido, el Spectrum activa los bits a 1 con el color de la tinta elegida, y los bits a 0 con el color del papel. Para nuestro ordenador sólo existirán dos colores, pero nosotros, al poseer una percepción óptica más o menos compleja, fu-



sionamos los dos tonos en uno solo, captando así la mezcla de los dos colores.

```
  10 REM ************************
  20 REM * J.M.MAYORAL SERRANO *
  30 REM ************************
  40 REM * PALETA     (C) 1986 *
  45 REM ************************
  50 POKE 23658,8
 100 PAPER 7: INK 9: BORDER 7
 150 CLS : PRINT  INK 1;"    127 COLORES PARA SPECTRU
M"
 160 REM [INV.] GRAFICOS DE USUARIO [TRUE]
 200 FOR N=0 TO 7
 210 READ M
 250 POKE USR "A"+N,M
 300 NEXT N
 500 GO SUB 1700
 510 REM [INV.]INTRO. CODIGO NUMERICO[TRUE]
 750 INPUT , LINE A$
 800 IF LEN A$<>2 THEN  GO TO 750
 850 IF CODE A$<48 OR CODE A$>55 OR CODE A$(2)<48 OR
CODE A$(2)>55 THEN  GO TO 750
 855 REM [INV.] ASIGNACION VALORES [TRUE]
 860 LET B=VAL A$(2)
 865 LET A=VAL A$(1)
 870 CLS : LET BRILLO=0
 880 GO SUB 1900
 890 LET BRILLO=1
 895 GO SUB 1900
 900 REM [INV.] MENSAJES FIN PROG. [TRUE]
 960 IF VAL A$(1)=0 AND VAL A$(2)=0 THEN  PRINT AT 8,
3;"ESTE COLOR SE OBTIENE CON:": PRINT AT 11,10;" EL C
OLOR ";A$(1);" ": GO TO 1100
1000 PRINT AT 3,3;"ESTE COLOR SE OBTIENE CON:": PRINT
 AT 5,2;"PRINT PAPER ";A;";INK ";B;";BRIGHT 0"
1010 PRINT AT 13,3;"ESTE COLOR SE OBTIENE CON:": PRIN
T AT 15,2;"PRINT PAPER ";A;";INK ";B;";BRIGHT 1"
1450 PRINT  INK 2;AT 21,2;" DESEAS CONTINUAR ?   "; I
NK 1;"(S/N)"
1460 LET K$=INKEY$
1500 IF K$="" THEN  GO TO 1460
1550 IF INKEY$="S" THEN  RUN
1600 IF INKEY$="N" THEN  GO TO 10000
1650 GO TO 1500
1700 REM [INV.]  PRESENTACION [TRUE]
1705 PRINT : PRINT
1710 FOR A=0 TO 7
1720 FOR B=0 TO 7
1730 PRINT A;B;
1740 PRINT  PAPER A; INK B;"AAAAAA";
1750 NEXT B
1760 NEXT A
1770 PRINT : PRINT
1780 PRINT "INTRODUCE EL CODIGO NUMERICO DEL"
1790 PRINT "        DEL COLOR ELEGIDO"
1800 RETURN
1900 REM [INV.] RELLENO DE PANTALLA [TRUE]
1910 FOR N=0 TO 9
1920 PRINT  PAPER A; INK B; BRIGHT BRILLO;"AAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAA"
1930 NEXT N
1940 RETURN
2000 DATA 170,85,170,85,170,85,170,85
```



Debemos tener en cuenta que ninguna de las mezclas puede ser utilizada para escribir sobre ella.

Nada más ejecutar el programa, aparece en la pantalla un menú con los diferentes tipos de colores a los cuales podemos acceder mediante un código numérico situado a la izquierda del tono en cuestión.

Este código numérico consta de dos cifras. La primera nos indica el color del papel y la segunda el de la tinta.

En el menú de presentación existen solamente 64 colores. Por cada tono elegido, nuestro Spectrum genera dos diferentes: uno sin brillo (**BRIGHT** 0) y otro con el brillo activado (**BRIGHT** 1).

Solamente existe un caso en donde el atributo «brillo» no ejerce ninguna influencia: cuando tanto el papel como la tinta son de color negro (paper 0; ink 0; código del programa=00).

Debemos tener muy en cuenta que ninguna de las mezclas que se obtienen de los ocho colores básicos no pueden ser utilizadas como «pizarra» para escribir o mensajes encima de estos, ya que, a causa del hardware, el Spectrum sólo puede mantener a un mismo tiempo dos colores básicos en una posición de caracter.

Si nuestro deseo es encontrar nuevos tonos de mezclas, lo único que debemos hacer es reformar la definición del gráfico definido por otra diferente a la que ya existe en el programa.

Así pues lo único que debemos cambiar es el contenido de la DATA en la última línea del programa.

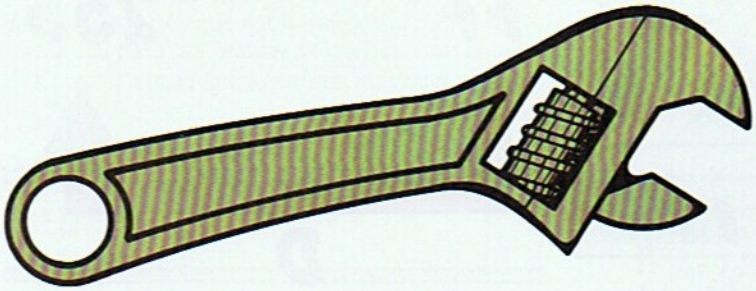
Ejemplo 1.
DATA 255,0,255,0,255,0,255,0
Ejemplo 2.
DATA 129,129,129,129,129,129,129,129,129

AYUDAS A LA PROGRAMACION

Dentro del vasto terreno del *software* podemos diferenciar varios campos en donde agrupar los programas. Uno de estos campos es el destinado a los programas de ayuda a la programación, o herramientas, que nos sirven para extraer mayor partido al ordenador y depurar nuestros programas.

En este breve muestrario de los ávidos programadores veremos, primeramente, un desensamblador, para obtener el listado en ensamblador y en código máquina de un programa en este lenguaje. El siguiente es un compilador, adecuado a los que desean velocidad en sus programas y no se atreven con el lenguaje máquina. ZXED es un programa que servirá de gran ayuda para los programadores al tener posibilidad de renumeración, saber la memoria libre... Para sacar más partido a su SPECTRUM está BETA-BASIC, que incorpora nuevos comandos, dando mayor potencia a la máquina. Por último, tenemos al «mini-lenguaje» DLAN que nos ofrecerá una sugerente presentación en la pantalla de nuestros textos.

Profundicemos ahora en estos programas que pueden ofrecer mucho en ayuda del programador asiduo.



Los programas herramientas, o de ayuda, sirven para depurar nuestros programas y sacar más partido a nuestro "micro".

EL DESENSAMBLADOR SPDE

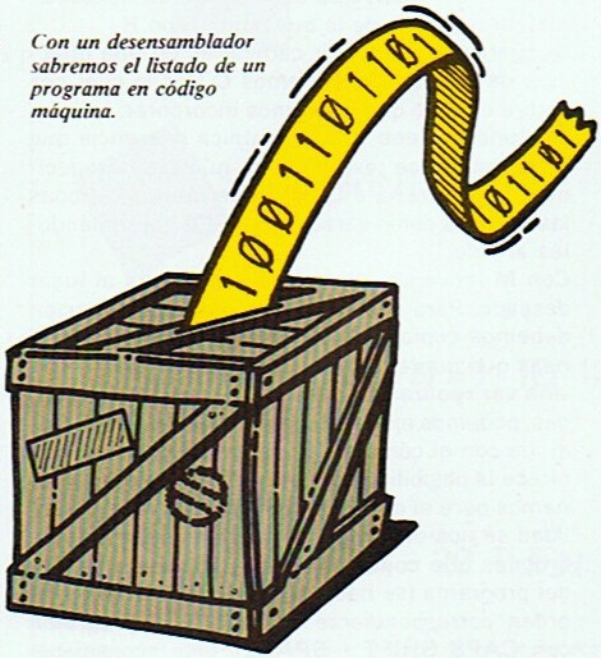
Ya hemos hablado en esta misma sección de lo que es un programa ensamblador y su utilidad. Sin embargo, también disponemos de la herramienta de programación contraria: un programa desensamblador.

Este tipo de programas nos sirve para obtener el listado en lenguaje máquina y en ensamblador de un programa en código máquina. Así, los programas en este lenguaje dejan de ser una caja misteriosa al intentar descubrir cómo realizan sus funciones.

Para trabajar con SPDE debemos primeramente cargar el programa a desensamblar, y posteriormente el desensamblador en sí. Lo primero que nos aparece es el menú de opciones. Para ver las instrucciones (incluidas en el programa) basta pulsar I.

El comando **A** es el que nos ofrece el listado del programa en cuatro columnas. La primera de ellas es la dirección de memoria en donde se encuentra almacenando el comando, la segunda muestra el contenido de dicha dirección. Las otras dos se refieren al lenguaje ensamblador, hallándose en la tercera el nemónico (el comando) y en la cuarta los parámetros correspondientes a la tercera columna.

Cuando el listado no quepa en una pantalla (como ocurrirá la mayor parte de las veces) pode-



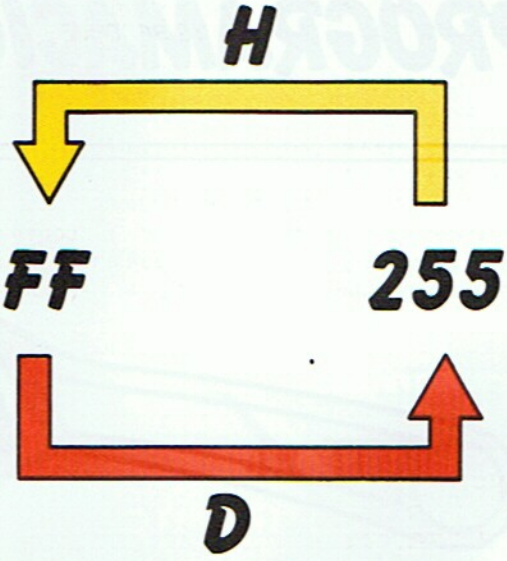
Con un desensamblador sabremos el listado de un programa en código máquina.

¡!

COMPILER no compila programas con números decimales, cadenas alfanuméricas ni tablas; otros comandos han sido transformados y las variables deben ir siempre en mayúsculas.

*

Un compilador es un programa que sirve para traducir nuestros programas BASIC a código máquina y obtener así una mayor velocidad de ejecución.



En DLAN, D transforma un número decimal en hexadecimal y H nos hace lo contrario.

mos irlo viendo si pulsamos las teclas del 1 al 9, apareciendo el número de líneas indicadas. Con **B** podremos ver el listado que se encuentra en las páginas anteriores, ya pasadas. Si pulsamos otras teclas diferentes avanzaremos, viendo nuevas páginas de listado.

Siempre que trabajamos en ensamblador o en lenguaje máquina tenemos necesidad de manejar números en hexadecimal obligándonos a realizar operaciones continuamente para hallar el equivalente. Con SPDE podemos efectuar la traducción directamente de hexadecimal a decimal y viceversa. **D**, tras introducir un número en decimal, nos devuelve su equivalente en hexadecimal, al contrario de lo que ocurre con **H**.

Si queremos introducir caracteres en una dirección de memoria pulsaremos **C** tras lo cual nos pedirá el texto que queremos incorporar. La misma función tiene **S**, con la única diferencia que no se introduce texto sino un número hexadecimal. La contraria a **C** es **F** que interpreta todas las órdenes como caracteres ASCII imprimiéndoles al lado.

Con **M** movemos bloques del programa al lugar deseado. Para ello SPDE nos pide qué dirección debemos copiar y la longitud, el número de líneas que queremos desplazar.

Una vez realizadas todas las operaciones deseadas, podemos ejecutar el programa de código máquina con el comando **E**, sin embargo no se nos ofrece la posibilidad de grabarlo en cinta; sí la tenemos para el desensamblador. Esta rara posibilidad se nos ofrece por si queremos eliminar las órdenes que contienen las instrucciones dentro del programa (se hace escribiendo el número de orden correspondiente tras cortar el programa con **CAPS SHIFT + SPACE**), algo aconsejable una vez se haya adquirido práctica con él, puesto que así disponemos de más memoria libre para desensamblar un programa.

Una deficiencia que presenta es la ausencia de una salida del listado por impresora obligándonos a trabajar con la pantalla con las limitaciones que ésto presenta.

EL COMPILADOR

Posiblemente alguna vez nos hayamos desesperado al observar la tremenda lentitud que tiene el BASIC en la ejecución de algunos programas largos y complicados. Es evidente que la única solución que le queda es recurrir al código máquina, con lo intrincado que resulta este camino. ¿No sería posible, entonces, que una vez que tengamos el programa BASIC hecho nos lo tradujera a código máquina y se ejecutase como tal? Es como si tuviéramos un programa compilador para BASIC (al igual que existen para otros lenguajes, como el PASCAL). Precisamente es lo que presentamos ahora, un programa que compilará (traducirá) nuestro programa BASIC al lenguaje má-

Hay comandos que COMPILER no puede traducir y otros que los transforma.

¡!

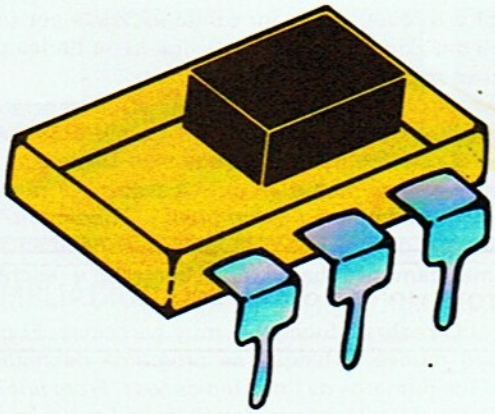
Con ZXED podemos, entre otras cosas, renumerar un programa y hacer que aparezca el número de la línea que estamos introduciendo de forma automática.

*

DLAN es un "mini-lenguaje" de programación, con grandes posibilidades gráficas, que permite una serie de opciones para presentación de textos en la pantalla.

*

La diferencia entre mover un bloque de programa y copiarlo está en que lo primero elimina el bloque en el lugar original y lo traslada al indicado; lo segundo, se mantiene el bloque en el lugar original copiándose donde fue indicado.



Podemos saber directamente la memoria que nos queda libre en la máquina.

quina y lo ejecutará como tal. El compilador que nos ocupa lo distribuye INVESTRONICA y es el COMPILER de *SOFTEK*.
Para trabajar con el COMPILER primeramente le debemos cargar, y una vez realizado podemos empezar ya a programar en BASIC. Antes de la compilación hay que comprobar muy bien el programa para evitar cometer alguna equivocación ya que el compilador tiene muy pocos mensajes de error. De hecho, es aconsejable grabar el programa en BASIC y después compilarlo.
Aunque suponga una maravilla tener un programa de este tipo, puesto que nos evita aprender el código máquina, también tiene sus limitaciones. Así, COMPILER no es capaz de compilar programas que incluyan números decimales, y por tanto no es aconsejable el uso de funciones que puedan producirlos, como es **SIN, COS, TAN, SQR, ↑**. Tampoco acepta cadenas alfanuméricas ni tablas (**DIM**). Además, muchos comandos serán interpretados de forma diferente a la usual, algunos para reforzar la función y otros para disminuirla. Así, no es necesario definir una variable antes de usarla, pero los **GO TO** y **GO SUB** no aceptan una variable ni una expresión. Sólo mencionar otro pequeño detalle: las variables deben ir siempre en mayúsculas. Antes de trabajar con el compilador es aconsejable leerse las instrucciones para conocer estas pequeñas diferencias.
Para entender mejor las ventajas que supone tener un compilador, vamos a teclear el siguiente programa:

```
10 FOR N=1 TO 100
20 PRINT AT 1,1;N
30 NEXT N
```

Una vez tecleado y comprobado que no hay ningún error y que cumple todas las condiciones que exige COMPILER podemos ya compilarlo tecleando **RANDOMIZE USR 49152**. Si no hay ningún problema nos aparecerá el mensaje:

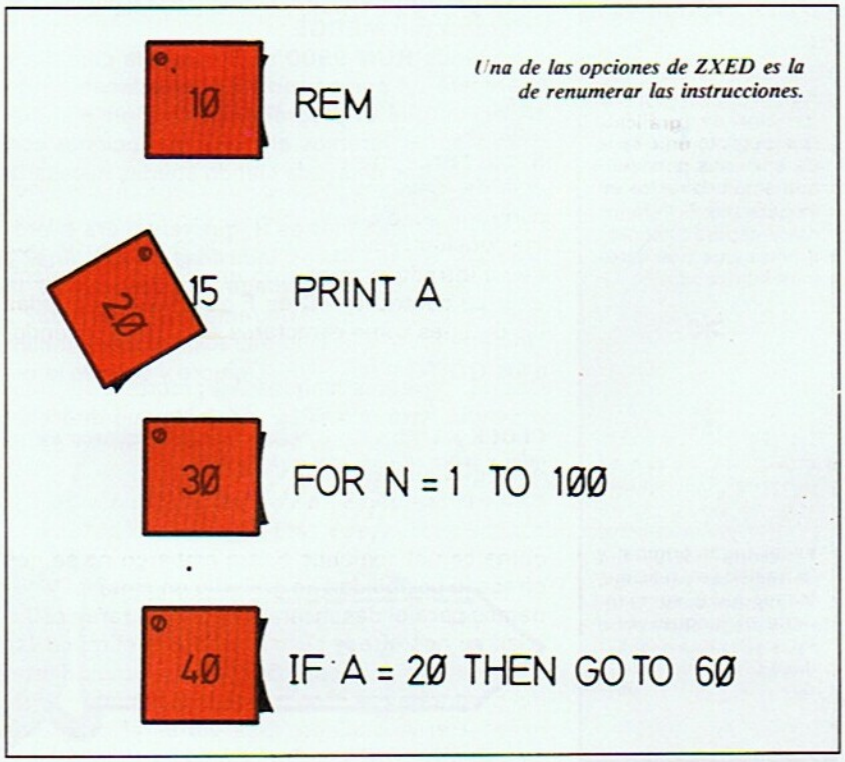
1>PROPERTY OF SOFTEK
ALL COMERCIAL RIGHTS RESERVED

Si lo ha ejecutado antes de compilarlo verá que el programa tarda 19 segundos en terminar. Pruébelo una vez compilado, haga **RUN** normalmente, y el tiempo que tarda ahora es de unos tres segundos. Sorprendente, ¿verdad?
Creo que queda clara la ventaja de un compilador, aunque tenga algunos inconvenientes. De todas maneras hay que estar atentos al mercado informático para las nuevas versiones y ampliaciones, como se asegura en las instrucciones del COMPILER.

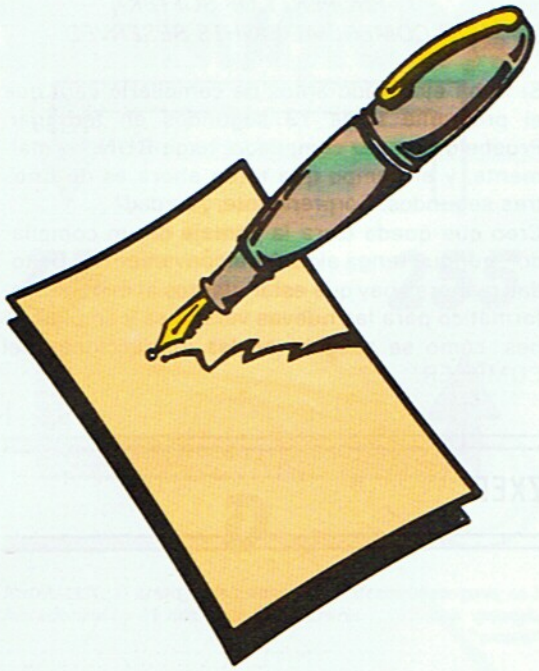
ZXED

Los aficionados a realizar sus propios programas quizá encuentren que el SPECTRUM no les da ninguna facilidad para realizar reformas según se va escribiendo, pero todo se complica más cuando el programa que tenemos entre manos es muy largo y necesitamos, por ejemplo, renumerar las líneas o cambiar de nombre a alguna variable.
Esta deficiencia la viene a suplir ZXED, un com-

Con el compilador no necesitamos saber código máquina, puesto que él mismo se encarga de traducir nuestro programa BASIC.



Una de las opciones de ZXED es la de renumerar las instrucciones.



ZXED nos permite copiar fragmentos de programa en el punto donde indiquemos.

pletísimo programa de ayuda a la programación, que nos sacará de más de un apuro a la hora del tecleo.

Para empezar a trabajar con él lo cargaremos con **LOAD**, como un programa normal. Pero si el programa a depurar lo tenemos en cinta, entonces se cargará primero ZXED con **LOAD** y después el programa con **MERGE**.

Ejecutando **RUN 9900** se limpiará la pantalla y aparecerá un cursor con "T" parpadenate, indicándonos que el programa ya está en marcha. Apretando H veremos el menú de opciones con unas palabras de ayuda siendo apenas necesaria explicación alguna.

Una de las más útiles es **R**, que renumera el programa entre las líneas indicadas *("From line?", "To line?")*, pudiendo elegir el comienzo de la nueva numeración y el incremento *("New base?", "Increment?")*. Esta renumeración afecta también a los **GO TO** y **GO SUB**, siempre y cuando la orden a la cual acudir no venga indicada por una variable. Si así fuera, ZXED nos avisa de las órdenes que no ha podido renumerar.

Cuando estamos tecleando un programa largo podemos evitarnos escribir el número de línea con el comando **S**, el cual nos pide desde cual línea y el incremento en que aparecerá la numeración automática *("From line?", "Increment?")*. Para que deje de aparecer el número de línea automáticamente basta con borrarlo y escribir **STOP**.

M y **C** realizan funciones muy parecidas. El primero mueve el bloque de programa delimitado por los números de línea indicados *("From line?", "To line?")* y lo coloca detrás de la línea que digamos *("Insert after?")*, eliminándole de su situación original. **C** copia un bloque de programa con las mismas condiciones que **M**, aquí, como es lógico, el bloque original no es eliminado. En ambos casos, una vez realizada la función nos indica dónde ha sido insertado el bloque en cuestión *("...To...Inserted")*, haciéndolo siempre con números de línea seguidos, por tanto es necesario después renumerar el programa.

Ya nos habremos percatado de lo fastidioso que resulta, al realizar una serie de cambios en un programa, ir borrando un conjunto de líneas que sobran una por una. Con ZXED tenemos la facilidad de borrar todo un bloque de una manera muy sencilla. Para ello tenemos **D**, que nos pide la primera línea a borrar *("From line?")* y la última *("To line?")*.

Podemos saber rápidamente la memoria que nos queda libre para programar con sólo pulsar **B**, proporcionando el resultado en *bytes*. Igualmente podemos saber en qué dirección de memoria se encuentra almacenada una línea con pulsar **L**. Otras dos opciones muy parecidas son **A** y **P**. Aquella cambia un carácter por el que nosotros le digamos *("...To be...")* dentro del margen de líneas que indiquemos, y el segundo comando nos escribe en las líneas que mencionemos el carácter deseado. Con **A**, una vez realizado el cambio nos escribe los números de orden en donde se hizo. Si no deseamos ver dichos números basta con pulsar **V** apareciendo el mensaje *Verify Off*. Por último, cuando deseemos grabar en cinta el programa que estamos realizando es necesario previamente borrar el ZXED eliminando las líneas 9900 a 9914 una a una, puesto que no permite el autoborrado con **D**.

¡!

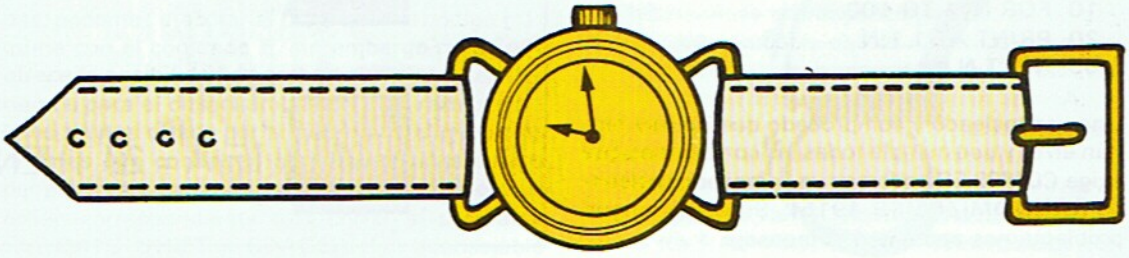
BETA-BASIC es una ampliación al lenguaje BASIC del SPECTRUM, que aumenta la potencia de programación al poder manejar comandos que aparecen en ordenadores muy superiores.

*

Los programas de ayuda a la programación nos sirven para sacar más partido al ordenador y depurar los programas que nos estemos fabricando.

*

Tenemos posibilidad de manejar ventanas, definir su color interior y elegir entre ocho tipos de contornos distintos.



CLOCK y TIMES nos da la posibilidad de manejar un reloj y saber en todo momento la hora.

A A A A A

AMPLIACION DEL BASIC

Para los expertos programadores quizás el BASIC del SPECTRUM se quede pequeño al faltarle una serie de instrucciones que en ciertas ocasiones pueden ser necesarias. Esto sólo tiene dos soluciones: la primera es la de cambiar de máquina, evidentemente la mejor pero la más costosa. La otra es la de obtener una ampliación del lenguaje BASIC para nuestra máquina. Esto último es lo que nos ofrece VENTAMATIC con el BETA-BASIC.

Este popular programa, del que ya han surgido múltiples versiones, nos da la posibilidad de tener unos comandos nuevos incorporados, y algunos de los existentes transformados, en nuestro ordenador, ofreciéndonos unas posibilidades que no tienen otros aparatos de precio más elevado. Ocupa unos 9 K., quedando por tanto 32 K. libres para la programación.

Comentaremos aquí algunos de los muchos comandos y novedades que incorpora:

— **BREAK**: No es un comando. Es el **BREAK** normal del SPECTRUM pero aumentada su potencia al poder cortar un programa en código máquina.

— **CLOCK, TIME$**: Controlan un reloj interno, que nosotros ponemos en hora, y que tiene posibilidad de alarma. Cada vez que lo requiramos podemos visualizarla.

— **ELSE**: Es el complemento a **IF...THEN**. Cada vez que una condición se cumple se ejecuta lo que se encuentra a continuación de **THEN**, si no se cumple, saltará y el programa seguirá después de **ELSE**.

— **RENUM**: Renumera los números de línea del programa con que estemos trabajando o del bloque de programa indicado. Afecta también a **GO TO, GO SUB, RUN...**

— **AUTO**: Nos va dando, automáticamente, el número de la línea que estamos introduciendo.

— **BIN$, DEC, HEX$**: Cambia un número al sistema de base que nosotros indiquemos. El primero, convierte un número en decimal a binario. El segundo, nos da como resultado un decimal tras introducir un número hexadecimal, y el tercero, transforma un decimal en hexadecimal.

— **MOD**: Da el resto de la división (es la función módulo).

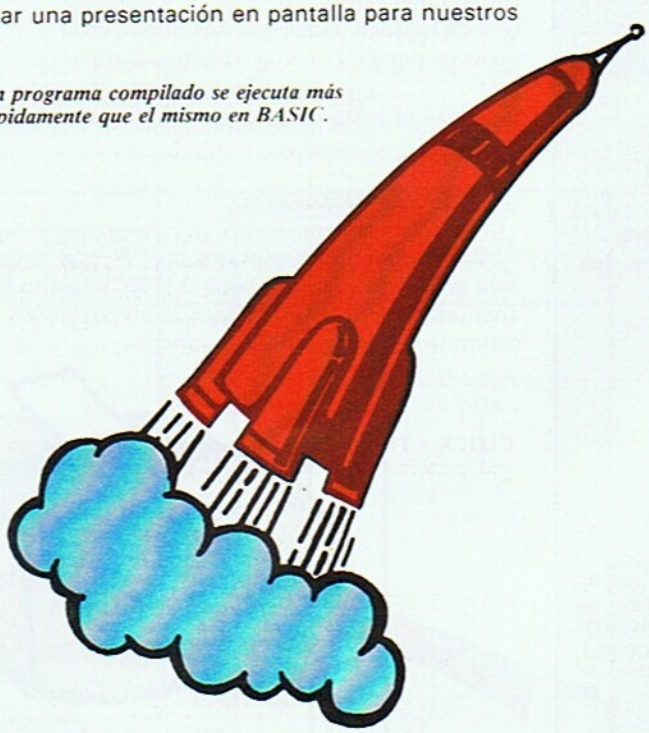
Estos, y más comandos que incorpora el BETA-BASIC, son cargados por medio de software en la RAM, por tanto al desconectar la alimentación de la máquina desaparecerá dicho programa. Igualmente, para ejecutar un programa en el que se hayan utilizado sentencias del BASIC ampliado será necesario primero cargar este programa.

DLAN es un programa que nos permite presentar textos de muy diferentes formas.

DLAN

Si alguna vez hemos tenido la necesidad de realizar una presentación en pantalla para nuestros

Un programa compilado se ejecuta más rápidamente que el mismo en BASIC.

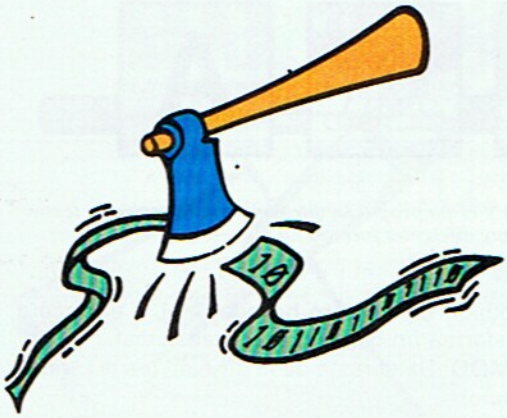


¡!

DLAN permite trabajar con 11 tipos de letras, imprimiendo el texto de izquierda a derecha (con =) y de derecha a izquierda (con <).

*

ELSE es el complemento a **IF...THEN**. Si la condición se cumple se ejecuta lo que está a continuación de **THEN**, en caso contrario, el programa seguirá después del **ELSE**.



El BREAK *del BETA-BASIC corta también los programas del código máquina.*

¡!

Un desensamblador sirve para obtener el listado en lenguaje ensamblador y en lenguaje máquina de un programa de este tipo.

*

Siempre que queramos ejecutar un programa que tenga comandos del BETA-BASIC será necesario cargar primero éste y luego el programa.

*

En el SPDE, **A** enseña el listado del programa en cuatro columnas: la dirección de memoria, contenido de la memoria, nemónicos y parámetros del nemónico.

textos verdaderamente sugerente, nos habremos visto en la necesidad de recurrir al código máquina. Sin embargo, ahora ya no es necesario, puesto que disponemos de DLAN *(Display Languaje*, en inglés), un lenguaje de programación que dispone de unos comandos para facilitarnos esta tarea.

Los comandos de DLAN deben ir escritos en instrucciones **REM** del BASIC y separados entre sí por ";", además de otro al final de la línea. Otra peculiaridad es la de no tener mensajes de error, por lo tanto, ante un parámetro incorrecto el programa lo ejecutará de una manera extraña, debiéndose repasar todo para averiguar donde se ha cometido el error.

Una de las facilidades que nos ofrece es la de manejar ventanas. Con ello podemos simular que tenemos una «mini-pantalla» y cuando se produzca un *scroll* el texto no se saldrá de la ventana definida.

10 REM W06090032;

Hemos definido una ventana, con **W**, que empezará en la línea 6 ("06") con 9 ("09") de alto. La anchura vendrá definida porque empezará en la columna 0 ("00") con 32 de ancho.

Si queremos hacer un reborde en el límite de la ventana, tenemos ocho donde elegir (de 1 a 8). Para ello ejecutaremos:

20 REM E8;

El color del interior lo elegiremos con **C** y el número del color correspondiente.

30 REM C4; F*;

En este caso tendremos una ventana de color verde. El comando que se encuentra a continuación (**F**) la llena del carácter escrito seguido a él, en este caso es "*".

Para escribir un texto disponemos de dos comandos. El primero es = que escribe lo que se encuentra a continuación de izquierda a derecha. Pero antes de imprimir es necesario especificar qué tipo de letra vamos a utilizar. Para ello disponemos de 11 tipos (del 1 al 9 más A y B). El comando que utilizamos es **T**.

40 REM T3;=BIENVENIDOS A;

En esta ocasión hemos elegido el tipo de letra 3 para escribir el texto "BIENVENIDOS A" (observemos que no es necesario entrecomillar el texto).

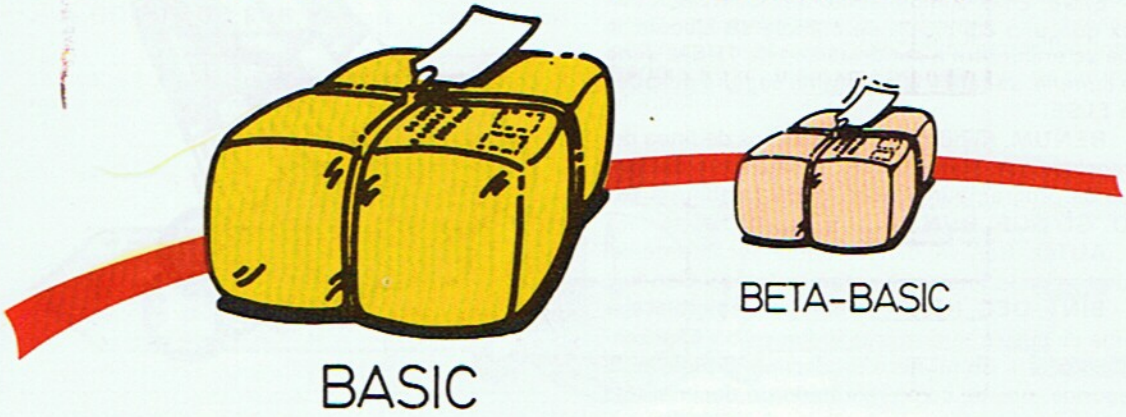
El segundo comando de impresión es < haciendo que el texto que se encuentre a continuación se imprima de derecha a izquierda.

50 REM T2;<DLAN;

En esta ocasión se imprime el texto "DLAN" con el tipo de letra 2.

Ahora es aconsejable probar el ejemplo aquí presentado como una muestra de lo que se puede hacer. Sin embargo, ofrece mucho más que es necesario conocer para obtener unos óptimos resultados con este lenguaje.

Con BETA-BASIC tenemos una ampliación al BASIC de SINCLAIR al tener nuevos comandos.



BIFURCACIONES

Durante los capítulos precedentes han ido apareciendo tablas resumidas de los diferentes grupos de instrucciones, donde entre otros datos de interés encontrábamos el efecto que determinada instrucción obraba sobre los indicadores F.
Pues bien, nuestro objetivo durante las próximas líneas será describir las instrucciones que modifican la secuencia de un programa. En ocasiones necesitaremos que ésta se rompa tan solo cuando determinada condición se cumpla, y lo «averiguaremos» analizando el contenido del bit especificado en el código de operación de la instrucción, dentro del registro F.

EL PROGRAM COUNTER (PC)

Por lo que hasta ahora sabemos, un programa no es otra cosa que una serie de bytes almacenados en la memoria de nuestro ordenador, que la CPU interpreta al objeto de llevar a cabo una determinada tarea.

En todo momento debe conocer la dirección a donde dirigirse y recavar la información, pues de otra manera el caos sería inevitable y la ejecución de cualquier rutina por sencilla que fuera, sería algo más que misión imposible.
Con la saludable intención de no armarse un fenomenal lío, el Z 80 conserva en su registro PC (contador de programa) la situación de la próxima instrucción a procesar y repite cíclicamente los siguientes pasos:
1) Lee el contenido del registro PC.
2) Se posiciona en la dirección de memoria indicada por éste.
3) Recoge allí el byte almacenado y averigua si es necesario leer octetos adicionales (recuerda que el formato de las instrucciones del Z 80 varía de 1 a 4 bytes).
4) Le añade al contador de programa PC el número de bytes de la instrucción actual, es decir, antes de procesarla, PC ya está señalando a la siguiente.
5) Finalmente, implementa la instrucción y vuelve al paso 1.
Evidentemente, siguiendo este sistema, los programas sólo podrían ejecutarse secuencialmente, es decir, recorriendo un byte tras otro hasta «chocar» con el final de la memoria. Pero tengamos en cuenta que aunque parezca que el ordenador no está haciendo nada mientras está conectado y no ejecutamos programa alguno, en el peor de los casos estará inmerso en alguna ruti-

¡!

Si el código máquina está todavía almacenado a partir de la dirección 23296 introduce como comando directo **GO TO 9996**. En caso contrario, **GO TO 9995** para recuperarlo del casete.

*

Señalemos cual es la primera línea a eliminar y cual la última. Tras ello, comprobaremos que estas han desaparecido de nuestro programa.

MNEMONICO	CODIGO MAQUINA	REGISTRO F 7 S	6 Z	5	4 H	3	2 P/v	1 N	0 C	N.º Bytes	CICLOS MAQ.	CICLOS RELOJ	NOTAS
CALL nn	1 1 0 0 1 1 0 1												
	n n n n n n n n												
	n n n n n n n n	•	•	•	•	•	•	•	•	3	5	17	ciclos si cc es falsa / A/B / ciclos si cc es verdadera
CALL cc,nn	1 1 x cc x 1 0 0												
	n n n n n n n n												
	n n n n n n n n	•	•	•	•	•	•	•	•	3	3/5	10/17	
RET	1 1 0 0 1 0 0 1	•	•	•	•	•	•	•	•	1	3	10	
RET cc	1 1 x cc x 0 0 0	•	•	•	•	•	•	•	•	1	1/3	5/11	

MNEMONICO	CODIDO MAQUINA								REGISTRO F								N.º Bytes	CICLOS		NOTAS
									7	6	5	4	3	2	1	0		MAQ.	RELOJ	
									S	Z		H		P/v	N	C				
RST p	1	1	x	t	x	1	1	1	•	•	•	•	•	•	•	•	1	3	11	

t	P	LLAMADA A LA RUTINA
000	00 H	INICIALIZACION DEL SISTEMA
001	08 H	GESTION DE ERRORES
010	10 H	IMPRESION DE UN CARACTER
011	18 H	RECOGIDA DE UN CARACTER
100	20 H	RECOGIDA DEL SIGUIENTE CARACTER
101	28 H	ENTRADA AL CALCULADOR
110	30 H	CREACION DE ESPACIO LIBRE
111	38 H	EXAMEN DEL TECLADO

¡!

Si la carga fue correcta procede a grabar el código máquina en casete mediante la orden **SAVE "C/M" CODE 23296,18.**

*

A continuación, tecleemos el programa DELETE cuidadosamente, respetando los números de instrucción asignados y grabémoslo con **SAVE "DELETE".**

*

Ahora ya podemos comenzar a trabajar con la rutina de borrado de líneas de programa. Supongamos que tenemos un programa en la memoria del cual deseas eliminar cierto bloque de instrucciones.

na de la ROM como, por ejemplo, la de análisis del teclado, en espera de alguna pulsación repitiéndola una y otra vez.
La clave está, por tanto, en el registro PC, y si pudiéramos modificar a nuestro antojo su contenido, sería posible alterar la secuencia de un programa de la misma manera que en BASIC lo hacemos mediante el comando **GO TO.**
Pero entre las instrucciones de carga de registros vistas anteriormente, no disponemos de ninguna del tipo LD PC,nn o algo parecido. Entonces, ¿son posibles las bifurcaciones en C/M? La respuesta es sí, y son las instrucciones de salto (*jump*, en inglés), las encargadas de llevarlas a cabo.

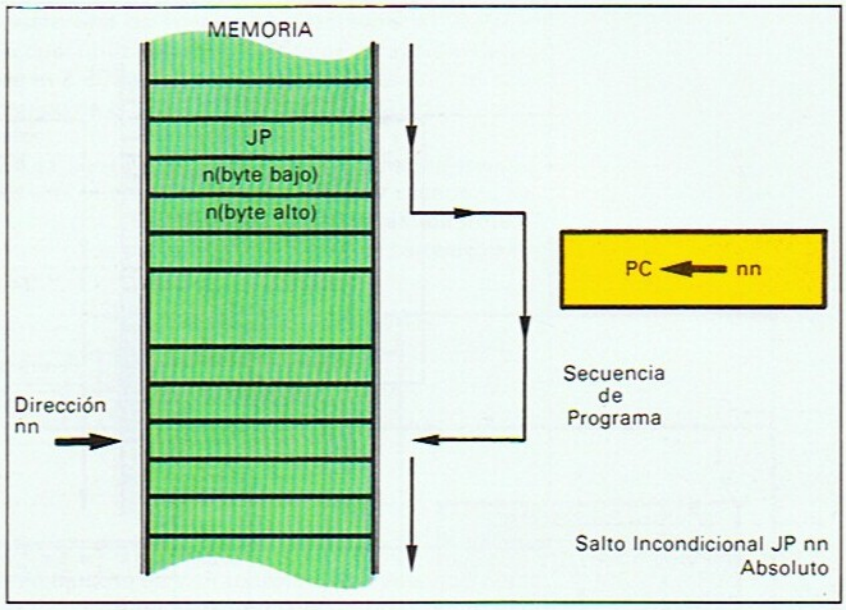
CAMBIOS DE SECUENCIA

Las instrucciones de salto provocan que el control del programa sea transferido a una dirección de memoria especificada en el propio código de operación. La manera de realizar la bifurcación depende del tipo de instrucción elegido, es decir, se pueden ejecutar saltos absolutos, relativos o por direccionamiento indirecto.
Tanto los absolutos como los relativos es posible efectuarlos de manera incondicional o tan solo cuando una determinada condición se verifique (saltos condicionales), mientras que los indirectos siempre son incondicionales.

SALTOS INCONDICIONALES

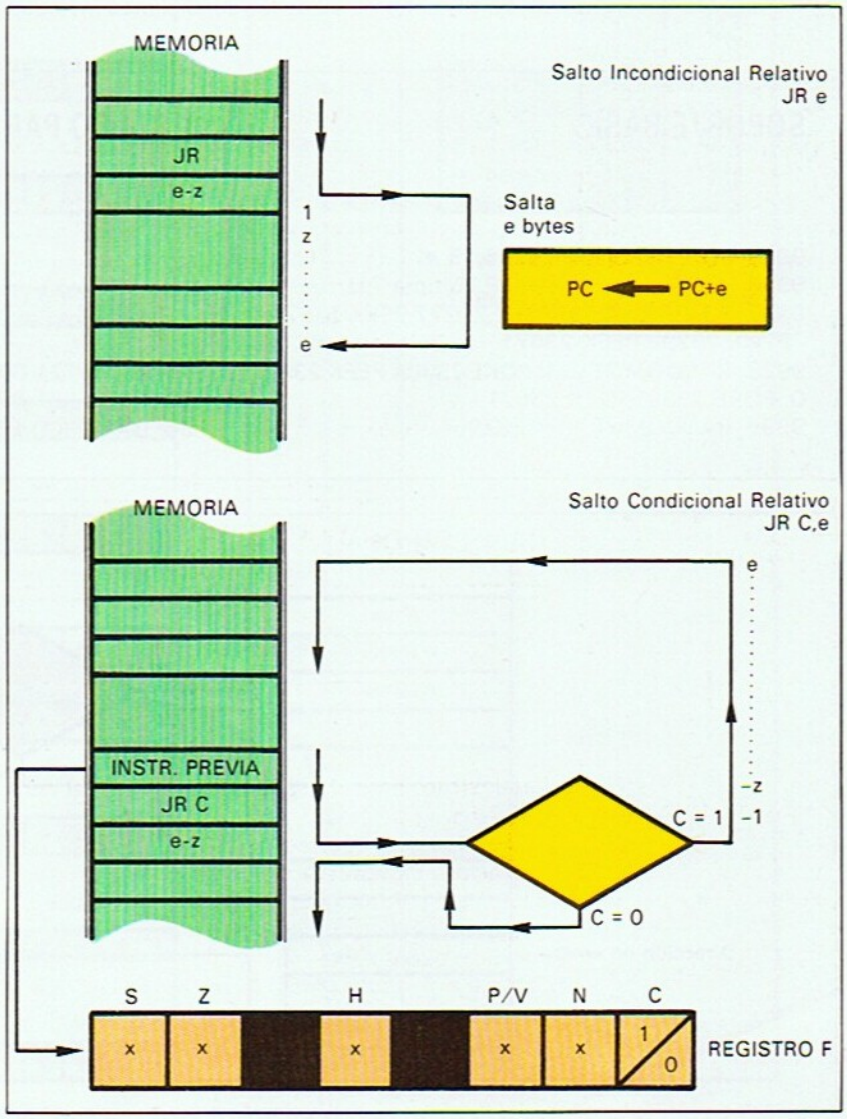
Cuando la CPU accede a la memoria y lee una instrucción de salto incondicional, tras decodificar el primer octeto, reconoce que se trata de una de este tipo. A continuación, cuando el salto es absoluto (JP nn), escruta los dos bytes siguientes donde encuentra la dirección a la cual saltar. Finalmente, carga en el registro PC este valor y continúa con la ejecución del programa a partir de esta última dirección.
El salto, aunque también incondicional, puede efectuarse de forma relativa, es decir, avanzando o retrocediendo en la memoria un determinado número de posiciones a partir de la señalada por el contador de programa PC.
En estos casos, la instrucción tan solo precisa de dos bytes, uno para el código de operación y otro para indicar el desplazamiento, el cual será un número en complemento a 2, y por tanto, en el rango de −128 a +127.
El mecanismo seguido por el Z 80 al implementarla se describe en la figura, y como podemos comprobar es de capital importancia tener presente que el desplazamiento se añade al PC suponiendo que éste se habría actualizado normalmente, como si la instrucción no hubiera sido de salto.
Dada esta circunstancia, los saltos permitidos al procesar una instrucción JR e, donde e es el des-

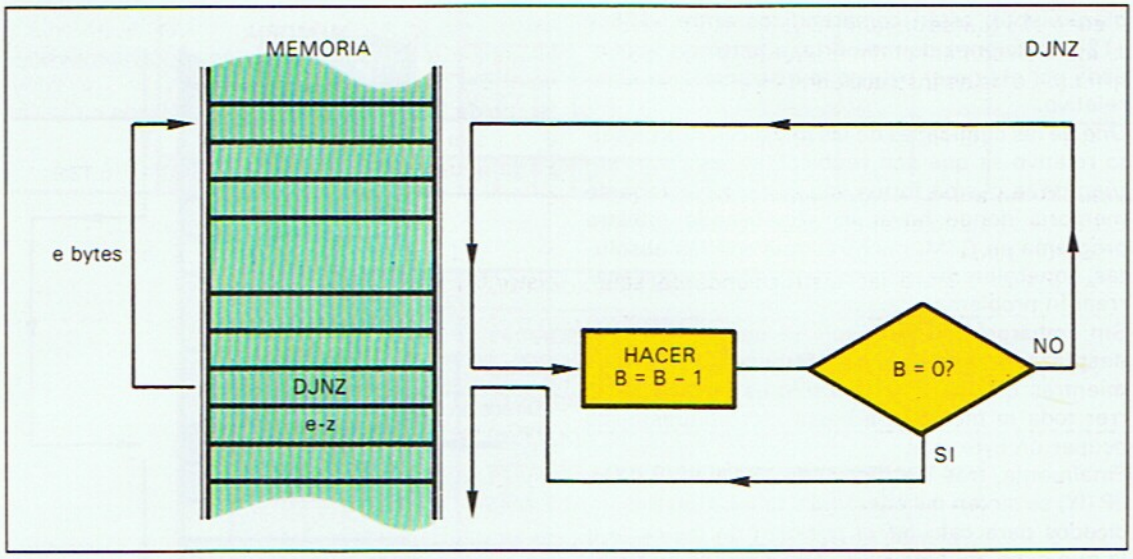
plazamiento, están comprendidos entre –126 y +129 posiciones de memoria, a partir de la ocupada por el primer byte de la instrucción de salto relativo.
Una de las cualidades de las instrucciones de salto relativo es que son reubicables, es decir, actúan de la misma forma sea cual sea la zona de memoria donde hayamos almacenado nuestro programa en C/M, mientras que con las absolutas, generalmente, no ocurre lo mismo y nos acarrearán problemas de adaptación.
Sin embargo, con ellas sólo se pueden efectuar desplazamientos en los márgenes señalados, mientras que los saltos absolutos pueden recorrer toda la memoria a costa, por supuesto, de ocupar un byte más.
Finalmente, tres instrucciones JP (HL), JP (IX) y JP (IY) se sirven del contenido de los registros indicados para calcular la posición de memoria a la que debe apuntar el PC. Obviamente en todas ellas, el desplazamiento se efectúa de forma absoluta.



LOS SALTOS CONDICIONALES

En el grupo anterior, cuando el microprocesador «tropezaba» en la memoria con una instrucción de salto, fuera cual fuera el estado del registro de indicadores F, SIEMPRE transfería el control del programa a la posición de memoria especificada en la instrucción.
Ahora analizaremos un nuevo tipo de instrucciones, las cuales ejecutan o no el salto (absoluta o relativamente, como antes) basándose en el estado de alguna de las banderas presentes en el registro F.
Para los indicadores de signo S, cero Z, paridad/desbordamiento P/V y arraste C es posible plantear la instrucción de salto absoluto para que en función de su contenido el programa bifurque o no.
JP NZ,nn: producirá salto a la dirección nn cuando el indicador de cero esté a 0.
JP Z,nn: salta si la bandera de cero está alzada. Aparentemente, existe un contrasentido en estas instrucciones, pero recordemos que el indicador de cero se sitúa a 1 cuando el resultado de la operación previa fue 0.
JP NC,nn: bifurca en el caso de comprobar que la bandera de acarreo está bajada.
JP C,nn: es la condición contraria a la anterior.
JP PO,nn: el salto es efectivo si en la última ope-





ración que afectó al indicador P/V se produjo paridad impar o no hubo sobrepasamiento.
JP PE,nn: salta de darse las condiciones inversas a las anteriores.
JP P,nn: tras escrutar el indicador de signo S, si éste contiene un 1 se produce el salto.
JP M,nn: es la opuesta de la anterior.
Los saltos condicionales relativos solamente

SOPORTE BASIC

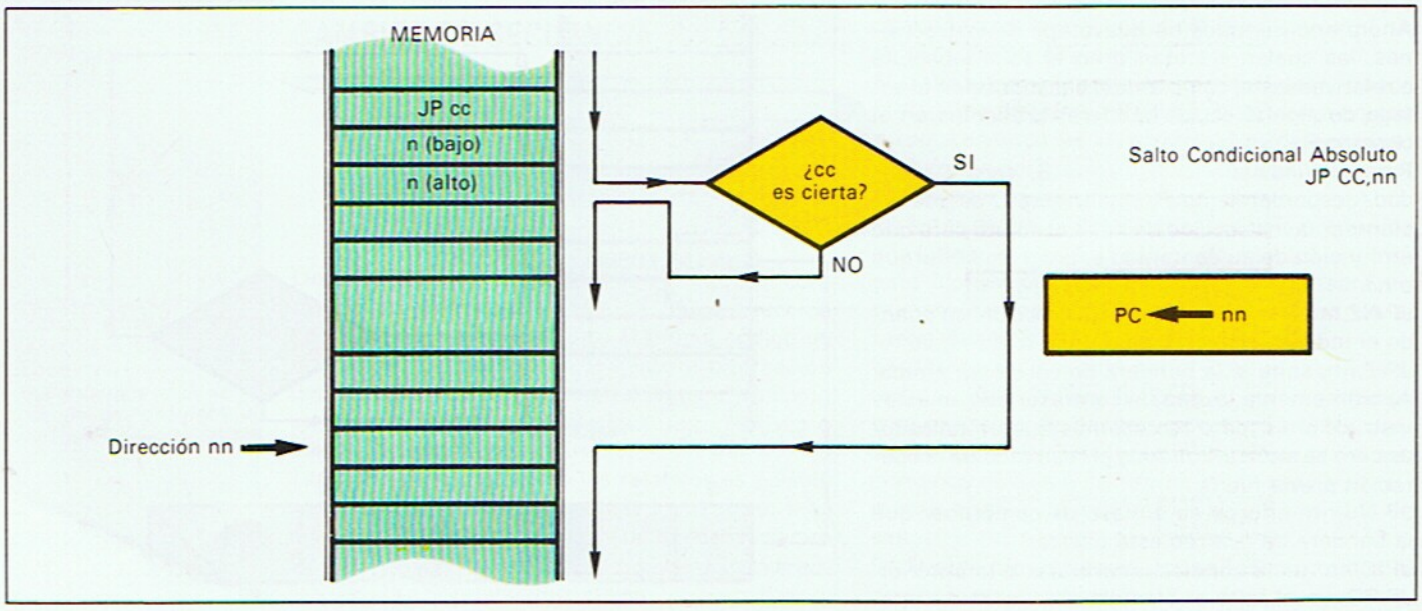
```
9995 LOAD ""CODE 23296,18
9996 INPUT "1a.- Linea ";P,"Ultima ";U
9997 RANDOMIZE P: POKE 23297,PEEK 23670:
  POKE 23298,PEEK 23671
9998 RANDOMIZE U+1: POKE 23304,PEEK 2367
0: POKE 23305,PEEK 23671
9999 RANDOMIZE USR 23296
```

LISTADO PARA CARGADOR

```
10 DATA "21 00 00 CD 6E 19 E5",602
20 DATA "21 00 00 CD 6E 19 D1",582
30 DATA "CD E5 10 C9 **",660
```

LISTADO ENSAMBLADOR

```
LD HL, PRIMERA
CALL 196E
PUSH HL
LD HL, ULTIMA+1
CALL 196E
POP DE
CALL 19E5
RET
```



pueden efectuarse dependiendo del estado de dos de los indicadores: el de acarreo y el de cero. El formato de estas instrucciones es el siguiente:

JR C,e JR NC,e JR Z,e JR NZ,e

donde como antes, e representa el byte de desplazamiento a sumar al PC, caso de cumplirse la condición fijada.
Para terminar, entre las instrucciones de salto existe una de funcionamiento especial (DJNZ,e) la cual utiliza el registro B como condición. Cuando el Z 80 la encuentra decrementa en 1 el contenido de B, comprueba si éste es 0 y de no serlo efectua el salto relativo.
En la figura puedes comprobar como su manejo es una buena técnica para efectuar repetidas pasadas por un grupo de instrucciones, durante el número de veces con que hayamos cargado el registro B.

MNEMONICO	CODIGO MAQUINA	REGISTRO F 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	N.º Bytes	CICLOS MAQ.	CICLOS RELOJ	NOTAS
JP nn	1 1 0 0 0 0 1 1												
	n n n n n n n n												
	n n n n n n n n	•	•	•	•	•	•	•	•	3	3	10	
JP cc,nn	1 1 x cc x 0 1 0												
	n n n n n n n n												
	n n n n n n n n	•	•	•	•	•	•	•	•	3	3	10	
JR e	0 0 0 1 1 0 0 0												
	x e z x	•	•	•	•	•	•	•	•	2	3	12	
JR C,e	0 0 1 1 1 0 0 0												
	x e z x	•	•	•	•	•	•	•	•	2	2/3	7/12	
JR NC,e	0 0 1 1 0 0 0 0												
	x e z x	•	•	•	•	•	•	•	•	2	2/3	7/12	
JR Z,e	0 0 1 0 1 0 0 0												
	x e z x	•	•	•	•	•	•	•	•	2	2/3	7/12	
JR NZ,e	0 0 1 0 0 0 0 0												
	x e z x	•	•	•	•	•	•	•	•	2	2/3	7/12	
JP (HL)	1 1 1 0 1 0 0 1	•	•	•	•	•	•	•	•	1	1	4	
JP (IX)	1 1 0 1 1 1 0 1												
	1 1 1 0 1 0 0 1	•	•	•	•	•	•	•	•	2	2	8	
JP (IY)	1 1 1 1 1 1 0 1												
	1 1 1 0 1 0 0 1	•	•	•	•	•	•	•	•	2	2	8	
DJNZ,e	0 0 0 1 0 0 0 0												
	x e z x	•	•	•	•	•	•	•	•	2	2/3	8/13	

CONDICION		CC
NZ	No cero	000
Z	Cero	001
NC	No arrastre	010
C	Arrastre	011
PO	Pondad impar	100
PE	Pondad par	101
P	Signo positivo	110
M	Signo negativo	111

• Indicador no afectado

A/B
- A: ciclos si NO se ha alcanzado la condición
- B: ciclos si se ha alcanzado la condición

e-z proporciona la dirección efectiva de PC+e, puesto que, PC es incrementado en 2 antes de la suma del desplazamiento e.

¡!

Para cargar la rutina DELETE en la memoria, utilicemos el cargador hexadecimal del capítulo 42, al que has de añadir las tres líneas **DATA**.

*

Efectua **RUN**, y cuando seas interrogado sobre «dirección de ubicación» contesta 23296.

LLAMADAS Y RETORNOS

Las llamadas (CALL) son instrucciones que modifican la ejecución secuencial de un programa, pero a diferencia de los saltos toman la precaución de anotar en el *stack* la dirección de memoria de la siguiente instrucción almacenada en la memoria tras ella.

Como podemos comprobar en la tabla correspondiente del capítulo destinado a la presentación del conjunto de ellas, (Saltos Y Llamadas) es factible ejecutarla incondicionalmente o sujeta a alguna restricción de las comentadas anteriormente.

Su funcionamiento es parecido al que en BASIC conseguimos mediante la sentencia **GO SUB**, es decir, cuando el Z 80 la encuentra, carga en el registro PC el contenido de los dos bytes siguientes determinando de esta manera la posición de memoria a donde transferir el control del programa, al objeto de ejecutar allí una determinada subrutina.

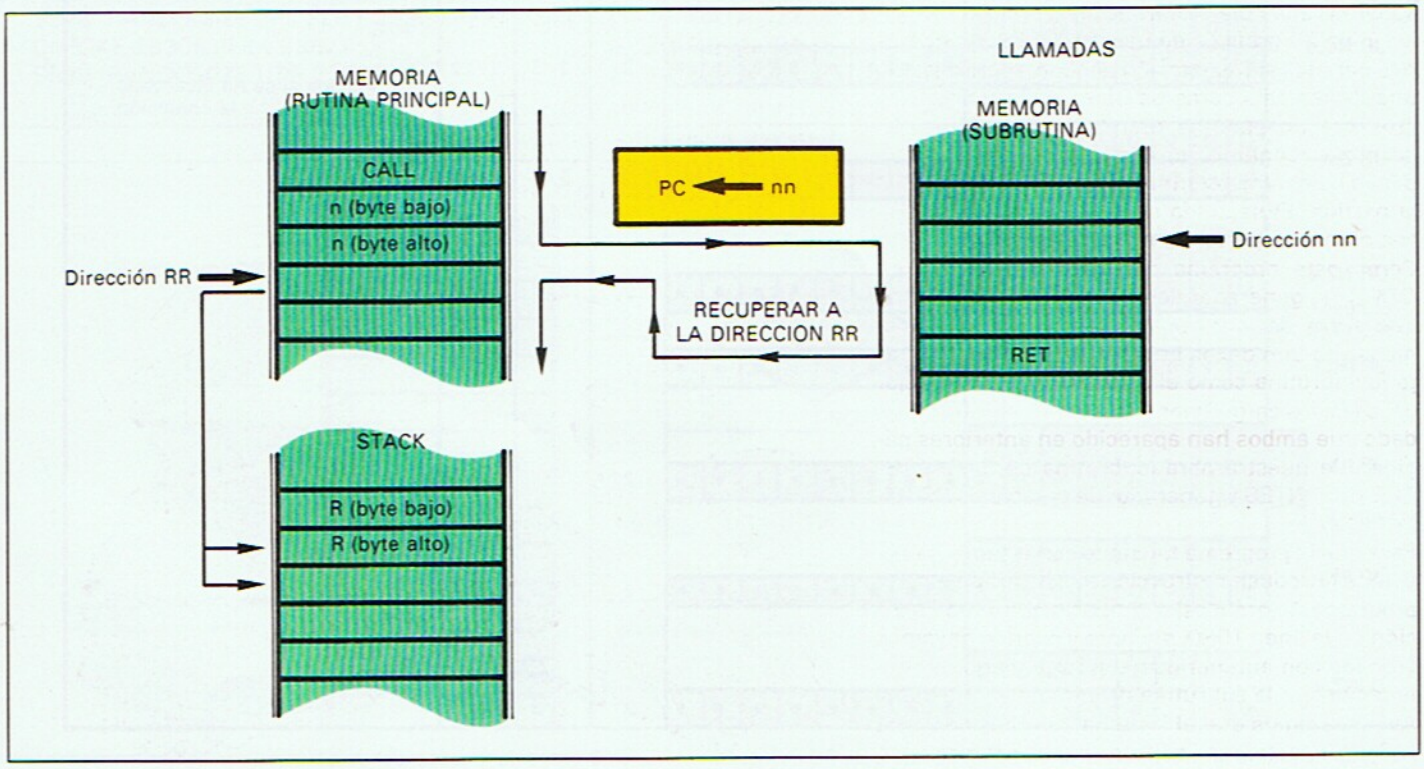
Pero antes de comenzar a procesar las instrucciones almacenadas a partir de dicha dirección, anota en el *stack* la posición de la instrucción siguiente a la llamada CALL. Tras ello, la subrutina es procesada hasta que aparece una instrucción de retorno incondicional RET o una condicionada RET cc, devolviéndose entonces el control a la rutina principal si se da la situación cc estipulada (mecanismo similar al RETURN del BASIC).

En estas circunstancias el microprocesador carga el PC con los dos bytes de lo alto del *stack* y regresa a la dirección especificada por éstos, los cuales, siempre que no manipulemos indebidamente la pila, han de ser los correspondientes a la siguiente posición de memoria tras la instrucción CALL.

Todas las instrucciones de llamada precisan de tres bytes para ser implementadas. Sin embargo, existen otro tipo de llamadas incondicionales de un byte, cuyo mnemónico es RST denominadas instrucciones de ReSTart.

Estas transfieren el control a la rutina almacenada a partir de la dirección especificada en su código de operación. En el caso del Spectrum, éstas se corresponden con ciertas subrutinas de interés incluidas en la ROM, cuyo cometido podemos encontrarlo en la tabla adjunta.

Entre las posibilidades de las instrucciones de llamada se encuentra el aprovechar las utilidades programadas en la ROM del sistema. Hemos preparado una pequeña rutina DELETE, la cual con tan solo 18 bytes elimina, instantáneamente, el bloque de líneas de programa BASIC que nosotros le indiquemos. Por supuesto, no está protegida frente al mal uso y siempre debemos indicar en primer lugar el número de línea inferior.



CALIMOCHO

Calimocho es un juego divertido y ante todo muy entretenido. Con este simpático programa aprenderemos a controlar nuestra codicia. Ya sabéis: «La avaricia rompe el saco». ¡Suerte y... a controlarse!

EL PROGRAMA

Calimocho es un programa en el que ha predominado sobre todo la estética y el color. La utilización de la subrutina de caracteres gigantes de PSION COMPUTERS, junto con el uso alternativo de dos generadores de caracteres (uno el estándar del Spectrum, y otro introducido por nosotros en R.A.M.) han hecho posible tal «perfección» plástica.

Nuestro programa CALIMOCHO consta de dos subprogramas bien diferenciados. El primero genera los 20 gráficos de usuario que son necesarios para la representación del contenido de cada una de las seis caras de que consta un dado, y presenta en pantalla una tabla de puntuación para que tengamos una idea de los puntos que podemos obtener dentro del normal desarrollo de la partida. El segundo subprograma, a todas luces el más importante, es el juego en sí.

Como este programa posee una subrutina en C/M y un generador de caracteres, aconsejamos que antes de la introducción del BASIC tecleemos cada uno de los listados que componen tanto la subrutina como el generador, o en el mejor de los casos carguemos tanto la una como el otro, dado que ambos han aparecido en anteriores páginas de nuestra obra (subrutina de CARACTERES GIGANTES y generador de caracteres ROTULACION).

Para que el programa funcione correctamente debemos introducir instrucción a instrucción el subprograma 1 (gráficos) y grabarlo con autoejecución en la línea 1080, siempre y cuando hayamos grabado con anterioridad el código máquina perteneciente a la subrutina de caracteres gigantes. Así evitaremos que el sistema se «cuelgue». Una vez hecho esto, realizaremos similar proceso con el segundo subprograma, grabándolo con autoejecución en la línea 2360 y a continuación el juego de caracteres ROTULACION a partir de la dirección 51656.

EL CALIMOCHO

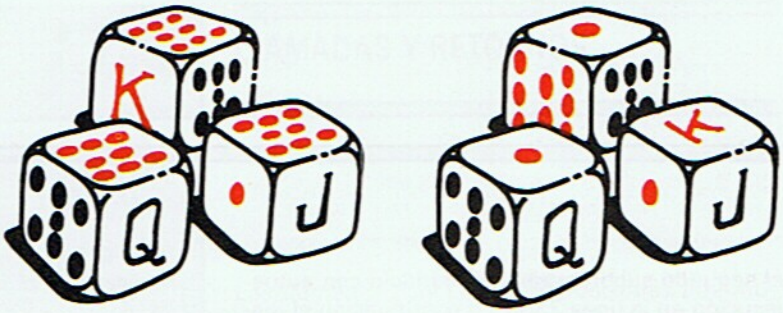
Las instrucciones para jugar al CALIMOCHO son las siguientes: se juega con tres dados de poker. En cada tirada, el Spectrum genera 3 valores aleatorios, (uno por cada dado) cada uno de los cuales se corresponderá con una representación en pantalla de los dados.

Sólo puntúan los «pitos» y los reyes (K), los primeros tienen un valor de 100 puntos mientras que los segundos valen 50. Tres «pitos» puntúan 1000 puntos, y tres reyes (KKK) 500.

Un jugador podrá plantarse cuando lo desee, pero si insiste en su afán de aumentar sus puntos personales, y en esa tirada no consigue puntuación, pierde los puntos parciales acumulados, pasando el turno al jugador siguiente.

De todas las jugadas que se pueden dar, la peor es el CALIMOCHO (3 rojos) es decir sacar en cada uno de los tres dados los ocho puntitos rojos. El Calimocho implica para el jugador que tenga la «fortuna» de conseguirlo, ¡empezar desde cero!

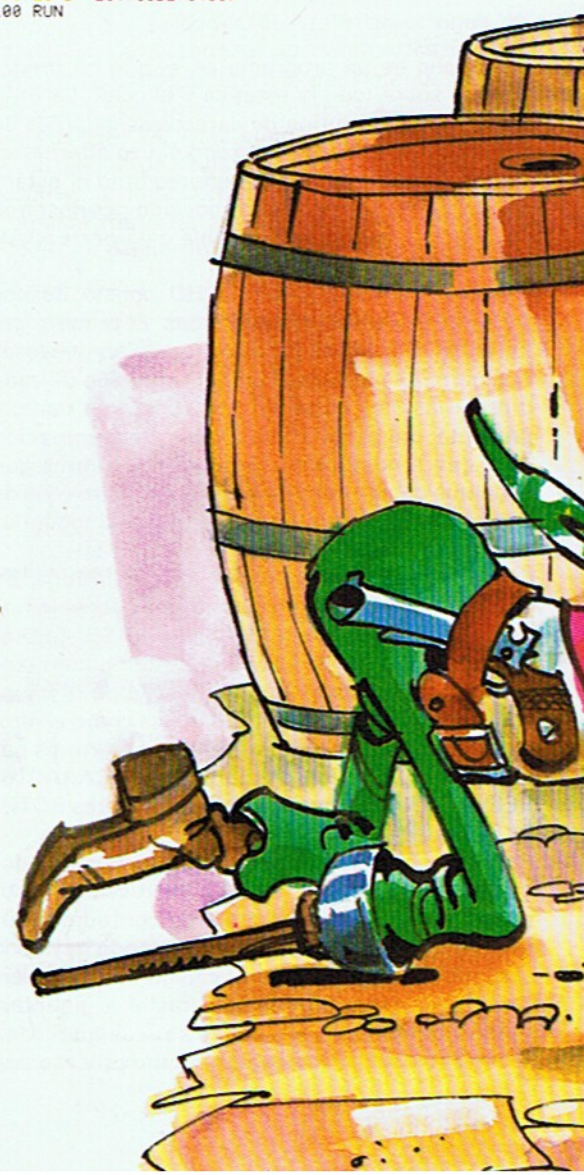
Con este simpático programa aprenderemos a controlar nuestra codicia; tengamos en cuenta que la avaricia rompe el saco.



En el CALIMOCHO utilizaremos tres dados de poker. La peor jugada es el CALIMOCHO: sacar en cada uno de los tres dados los ocho puntitos rojos, lo cual equivale a empezar desde cero.

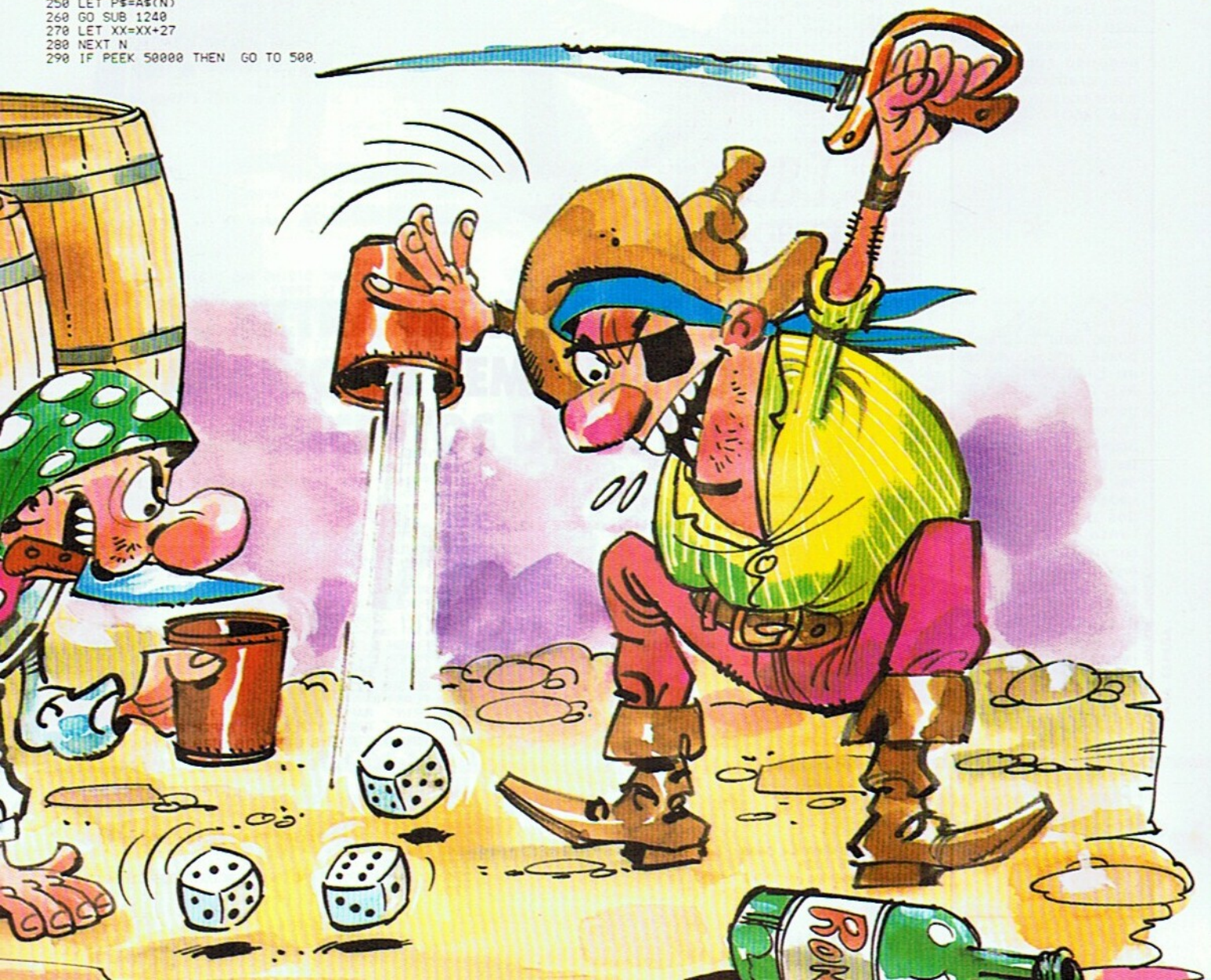
SUBPROGRAMA 1

```
   1 REM ************************          * J.M.MAYORA
L SERRANO *                    **********************          *
 CALIMOCHO  (C) 1985 *          **********************
*
   5 REM * PROGRAMA PARA 48 K. *          ************
***********
  10 LET YY=50
  20 LET FILA=4
  30 DATA 3,15,63,63,127,127,255,255
  40 DATA 192,240,252,252,254,254,255,255
  50 DATA 255,255,127,127,63,63,15,3
  60 DATA 255,255,254,254,252,252,240,192
  70 DATA 3,3,3,0,0,0,0,0
  80 DATA 254,254,254,112,112,112,112,112
  90 DATA 0,112,112,112,120,63,63,31
 100 DATA 112,112,112,112,240,224,224,192
 110 DATA 15,63,63,120,112,112,112,112
 120 DATA 192,240,240,120,56,56,56,56
 130 DATA 119,115,113,120,63,63,15,0
 140 DATA 184,216,232,240,120,184,188,60
 150 DATA 97,243,243,97,0,0,12,30
 160 DATA 134,207,207,134,0,0,48,120
 170 DATA 30,12,0,0,97,243,243,97
 180 DATA 120,48,0,0,134,207,207,134
 190 DATA 252,252,252,112,112,113,115,119
 200 DATA 127,127,127,60,248,224,192,128
 210 DATA 127,127,124,112,112,252,252,252
 220 DATA 192,224,240,120,60,127,127,127
 230 DATA 6,12,"100 Puntos.",145,6,13,"50 Puntos.",15
3
 240 DATA 12,5,"1000 Puntos.",137,12,6,"500 Puntos.",
145
 250 DATA 12,7,"CALIMOCHO.",153
 260 REM [INV.]  GRAFICOS DE USUARIO [TRUE]
 270 LET A$="ABCDEFGHIJKLMNOPQRST"
 280 FOR G=1 TO LEN A$
 290 FOR F=0 TO 7
 300 READ A
 310 POKE USR A$(G)+F,A
 320 NEXT F
 330 NEXT G
 340 PAPER 0: BORDER 0: INK 9
 350 CLS
 360 LET C=3
 370 LET P$="CALIMOCHO"
 380 LET YY=0
 390 LET XS=3: LET YS=3
 400 INK 1: PAPER 7
 410 GO SUB 970
 420 LET P$="TABLERO DE PUNTUACIONES"
 430 LET YY=25
 440 LET XS=1: LET YS=2
 450 INK 5: PAPER 0
 460 GO SUB 970
 470 BRIGHT 1: INK 2
 480 PRINT AT 6,C;"AB"
 490 PRINT AT 7,C;"CD"
 500 GO SUB 650
 510 PRINT AT 9,C;"QR"
 520 PRINT AT 10,C;"ST"
 530 GO SUB 650
 540 PRINT AT 12,C;"AB AB AB"
 550 PRINT AT 13,C;"CD CD CD"
 560 GO SUB 650
 570 PRINT AT 15,C;"QR QR QR"
 580 PRINT AT 16,C;"ST ST ST"
 590 GO SUB 650
 600 PRINT AT 18,C;"MN MN MN"
 610 PRINT AT 19,C;"OP OP OP"
 620 GO SUB 650
 630 GO TO 710
 640 REM [INV.]  PRESENTACION [TRUE]
 650 BEEP .2,40: INK 6
 660 READ COL,P,P$,XX
 670 GO SUB 900
 680 GO SUB 980
 690 INK 2
 700 RETURN
 710 FOR G=0 TO 1
 720 FOR F=2 TO 7
 730 INK F
 740 BRIGHT 0
 750 GO SUB 980
 760 NEXT F
 770 NEXT G
 780 FOR G=1 TO 3
 790 FOR N=1 TO 10
 800 BEEP .02,N+29
 810 NEXT N
 820 NEXT G
 830 BEEP .5,40
 840 CLS
 850 LET P$="Pulsa  PLAY  en el cassette"
 860 LET yy=80
 870 GO SUB 970
 880 LOAD "CALIMOCHO2"
 890 REM [INV.] RUTINA 1 [TRUE]
 900 LET YY=YY+24: LET FILA=FILA+3
 910 FOR Q=0 TO P-1
 920 PRINT  INK 9;AT FILA,COL+Q;"."
 930 BEEP .02,10
 940 NEXT Q
 950 RETURN
 960 REM [INV.]SBR. CARACT. GIGANTES[TRUE]
 970 LET xx=(256-8*xs*LEN p$)/2
 980 LET i=23306
 990 POKE i,xx: POKE i+1,yy: POKE i+2,xs: POKE i+3,ys
: POKE i+4,8
1000 LET i=i+4: LET w=LEN p$
1010 FOR q=1 TO w
1020 POKE i+q,CODE p$(q)
1030 NEXT q
1040 POKE i+w+1,255
1050 LET w=USR 64869
1060 RETURN
1070 REM [INV.] AUTO-EJEC. OBLIGATORIA[TRUE]
1080 CLEAR 64868
1090 LOAD "LIT"CODE 64869
1100 RUN
```



SUBPROGRAMA 2

```
   1 REM ************************          * J.M.MAYORA
L SERRANO *          ************************          *
 CALIMOCHO  (C) 1985 *          ********************
*
   5 REM * PROGRAMA PARA 48 K. *          ************
***********
  10 PAPER 0: BORDER 0: CLS
  20 GO SUB 1330
  30 LET C$=''
  40 LET D$=''
  50 LET FIN=0
  60 POKE 23658,8
  70 DIM P(3)
  80 DIM T(3)
  90 DIM A(3)
 100 DIM D(3)
 110 LET PUNTOS=0
 120 LET TINTA=3
 130 PAPER 0: BORDER 1: INK 9
 140 CLS
 150 POKE 23606,200: POKE 23607,200
 160 GO SUB 1590
 170 LET A$='CALIMOCHO'
 180 LET XS=3: LET YS=6
 190 FOR F=0 TO 1
 200 LET YY=8*F*14+8: LET XX=8
 210 FOR N=1 TO LEN A$
 220 INK TINTA
 230 IF TINTA=6 THEN  LET TINTA=2
 240 LET TINTA=TINTA+1
 250 LET P$=A$(N)
 260 GO SUB 1240
 270 LET XX=XX+27
 280 NEXT N
 290 IF PEEK 50000 THEN  GO TO 500
 300 LET YS=4
 310 NEXT F
 320 POKE 23606,0: POKE 23607,60
 330 LET P$='## R U N ##'
 340 LET YY=80: LET XS=2: LET YS=2
 350 GO SUB 1230
 360 LET P$='Pulsa una tecla'
 370 LET YY=180: LET XS=2: LET YS=1
 380 PAPER 1
 390 GO SUB 1230
 400 POKE 50000,1
 410 PRINT  PAPER 0; OVER 1; INK TINTA;AT 10,5;'
.
 420 PRINT  PAPER 0; OVER 1; INK TINTA;AT 11,5;'
.
 430 IF TINTA=7 THEN  LET TINTA=0
 440 LET TINTA=TINTA+1
 450 IF INKEY$='' THEN  GO TO 410
 460 LET P$='
.
 470 GO SUB 1230: PAPER 0
 480 GO SUB 1620
 490 GO SUB 1700
 500 GO SUB 1660
 510 POKE 23606,0: POKE 23607,60
 520 FOR N=1 TO 3
 530 PRINT AT 9,10*N-9; PAPER N+3; INK 9;'  MOCHO ';N
;'
.
 540 PRINT AT 11,10*N-6;'0000'
 550 NEXT N
 560 LET A$='MNMNEFIJQRAB'
 570 LET B$='OPOPGHKLSTCD'
 580 PRINT  INK 9; PAPER 0;AT 18,6;'PUNT. PARCIALES=
.
```



¡!

Para que el programa funcione correctamente debemos introducir instrucción a instrucción el subprograma 1 (gráficos) y grabarlo con autoejecución en la línea 1080, siempre y cuando hayamos grabado con anterioridad el código máquina perteneciente a la subrutina de caracteres gigantes. Una vez hecho esto, realizaremos similar proceso con el segundo subprograma, grabándolo con autoejecución en la línea 2360.

*

Como este programa posee una subrutina en C/M (CARACTERES GIGANTES) y un generador de caracteres (ROTULACION), aconsejamos que antes de la introducción del BASIC tecleemos cada uno de los listados que componen tanto la subrutina como el generador, o en el mejor de los casos carguemos tanto la una como el otro, dado que ambos han aparecido en anteriores páginas de nuestra obra.

```
 590 PRINT #0; INK 6; PAPER 1;' ~1~- JUGADA     ~P~-
PLANTARSE'
 600 POKE 23606,200: POKE 23607,200
 610 PRINT  INK 2; BRIGHT 1;AT 14,25;'-TOPE-'
 620 PRINT  INK 6;AT 16,26;TP
 630 POKE 23606,0: POKE 23607,60
 640 FOR T=1 TO 3
 650 GO SUB 1790
 660 GO SUB 1740
 670 PRINT  FLASH 1; OVER 1; PAPER T+3;AT 9,10*T-9;'
                                                    '
 680 LET K$=INKEY$
 690 IF K$='1' THEN  BEEP .05,40: GO TO 730
 700 IF K$='P' THEN  BEEP .5,40: GO TO 860
 710 RANDOMIZE
 720 BEEP .01,30: GO TO 680
 730 GO SUB 1790
 740 FOR D=1 TO 3
 750 LET D(D)=INT (RND*6)+1
 760 NEXT D
 770 GO SUB 1860
 780 GO SUB 1990
 790 IF FIN THEN  LET FIN=0: GO SUB 2170
 800 IF CAM=3 THEN  LET CAM=0: LET PUNTOS=0
 810 IF PUNTOS=0 THEN  BEEP .3,40: BEEP .6,20: GO SUB
2080: GO SUB 2120: GO SUB 2150: GO TO 830
 820 GO TO 680
 830 NEXT T
 840 IF T>=3 THEN  GO TO 640
 850 STOP
 860 FOR B=30 TO 45
 870 BEEP .03,B
 880 NEXT B
 890 LET E$=STR$ PUNTOS
 900 LET P(T)=P(T)+PUNTOS
 910 LET C1=INT ((10-LEN E$)/2)
 920 LET C1=C1+10*T-9
 930 PRINT  PAPER 0;;AT 11,10*T-9;'
                                    '
 940 PRINT  PAPER 0; INK 9;AT 11,C1;P(T)
 950 LET PUNTOS=0
 960 GO SUB 2080
 970 GO SUB 2150
 980 GO SUB 2120
 990 IF P(T)>=TP THEN  GO TO 1010
1000 GO TO 830
1010 PRINT  FLASH 1; INK 2; PAPER 7; OVER 1;AT 9,10*T
-9;'
                                                    '
1020 PRINT  FLASH 1; INK 7; PAPER 2; OVER 1;AT 11,10*
T-9;'
                                                    '
1030 POKE 23606,200
1040 POKE 23607,200
1050 LET P$='GANADOR-MOCHO '+STR$ T
1060 BRIGHT 1: PAPER 0
1070 FOR T=7 TO 3 STEP -1
1080 INK T
1090 GO SUB 1230
1100 NEXT T
1110 POKE 23606,0
1120 POKE 23607,60
1130 BRIGHT 0
1140 INK 5: PAPER 1
1150 LET YY=177: LET XS=1
1160 LET YS=2
1170 LET P$='-PULSA UNA TECLA PARA EMPEZAR-'
1180 GO SUB 1230
1190 LET K$=INKEY$
1200 IF K$='' THEN  GO TO 1190
1210 RUN
1220 STOP
1230 LET xx=(256-8*xs*LEN p$)/2
1240 LET i=23306
1250 POKE i,xx: POKE i+1,yx: POKE i+2,xs: POKE i+3,ys
: POKE i+4,8
1260 LET i=i+4: LET w=LEN p$
1270 FOR q=1 TO w
1280 POKE i+q,CODE p$(q)
1290 NEXT q
1300 POKE i+w+1,255
1310 LET w=USR 64869
1320 RETURN
1330 INK 7: POKE 23606,0: POKE 23607,60
1340 LET P$='TIENES QUE INTRODUCIR EL TOPE'
1350 LET YY=50
1360 LET XS=1: LET YS=2
1370 GO SUB 1230
1380 LET P$='NUMERICO AL QUE SE VA A AJUSTAR'
1390 LET YY=70
1400 GO SUB 1230
1410 LET YY=90
1420 LET P$='ESTA PARTIDA DE CALIMOCHO.'
1430 GO SUB 1230
1440 POKE 23606,200
1450 POKE 23607,200
1460 LET YY=110
1470 INK 5
1480 LET P$='MAX.= 5000       MIN.= 50000'
1490 GO SUB 1230
1500 POKE 23606,0: POKE 23607,60
1510 INPUT '          TOPE= ';TP
1520 IF TP<5000 OR TP>50000 THEN  GO TO 1510
1530 LET P$='TOPE-PARTIDA= '+STR$ TP+' PUNTOS.'
1540 INK 6
1550 LET YY=140
1560 GO SUB 1230
1570 FOR N=1 TO 300: NEXT N
1580 RETURN
1590 DRAW 255,0: DRAW 0,175
1600 DRAW -255,0: DRAW 0,-175
1610 RETURN
1620 FOR N=10 TO 20
1630 PRINT  PAPER 0;AT N,1;'
                                                    '
1640 NEXT N
1650 RETURN
1660 INK 9
1670 PLOT 5,168: DRAW 245,0
1680 DRAW 0,-57
1690 DRAW -245,0: DRAW 0,57
1700 FOR N=1 TO 6
1710 PRINT AT N,1; PAPER 0; OVER 1; INK N+1;'
                                                    '
1720 NEXT N
1730 RETURN
1740 LET P$=' Juega MOCHO '+STR$ T+' '
1750 PAPER T+3: INK 1
1760 LET YS=2: LET XS=2: LET YY=152:
1770 GO SUB 1230
1780 RETURN
1790 FOR Q=8 TO 20 STEP 6
1800 PRINT  PAPER 7; INK 6;AT 13,Q;'4337'
1810 PRINT  PAPER 7; INK 6;AT 14,Q;'5  5'
1820 PRINT  PAPER 7; INK 6;AT 15,Q;'5  5'
1830 PRINT  PAPER 7; INK 6;AT 16,Q;'1332'
1840 NEXT Q
1850 RETURN
1860 FOR D=1 TO 3
1870 LET D1=2*D(D)-1
1880 LET C$=A$(D1 TO D1+1)
1890 LET D$=B$(D1 TO D1+1)
1900 IF D(D)=1 OR D(D)=3 OR D(D)=4 THEN  LET TINTA=0:
 GO TO 1920
1910 LET TINTA=2
1920 LET Q=6*D+2
1930 PRINT  BRIGHT 1; PAPER 7; INK 6;AT 13,Q;'4337'
1940 PRINT  BRIGHT 1; PAPER 7; INK 6;AT 14,Q;'5'; INK
 TINTA;C$; INK 6;'5'
1950 PRINT  BRIGHT 1; PAPER 7; INK 6;AT 15,Q;'5'; INK
 TINTA;D$; INK 6;'5'
1960 PRINT  BRIGHT 1; PAPER 7; INK 6;AT 16,Q;'1332'
1970 NEXT D
1980 RETURN
1990 LET CAM=0
2000 IF D(1)=5 AND D(2)=5 AND D(3)=5 THEN  LET PUNTOS
=PUNTOS+500: GO TO 2080
2010 IF D(1)=6 AND D(2)=6 AND D(3)=6 THEN  LET PUNTOS
=PUNTOS+1000: GO TO 2080
2020 IF D(1)=2 AND D(2)=2 AND D(3)=2 THEN  LET P(T)=0
: LET PUNTOS=0: LET FIN=1: GO TO 2080
2030 FOR D=1 TO 3
2040 IF D(D)=5 THEN  LET PUNTOS=PUNTOS+50
2050 IF D(D)=6 THEN  LET PUNTOS=PUNTOS+100
2060 IF D(D)=1 OR D(D)=2 OR D(D)=3 OR D(D)=4 THEN  LE
T CAM=CAM+1
2070 NEXT D
2080 PRINT  INK 9; PAPER 0;AT 18,6;'PUNT. PARCIALES=
';PUNTOS;'
                                                    '
2090 RETURN
2100 PRINT  OVER 1; FLASH 0; PAPER T+3; INK 9;AT 9,10
*T-9;'
                                                    '
2110 RETURN
2120 LET P$='
                                                    '
2130 PAPER 0: GO SUB 1230
2140 RETURN
2150 PRINT  FLASH 0; OVER 1; PAPER T+3;AT 9,10*T-9;'
                                                    '
2160 RETURN
2170 POKE 23606,200
2180 POKE 23607,200
2190 LET P$='-  CALIMOCHO  -'
2200 FLASH 1
2210 PAPER 1
2220 INK 7
2230 GO SUB 1230
2240 FLASH 0
2250 PRINT  PAPER 0;AT 11,10*T-9;'  '; FLASH 1; INK 2
; PAPER 6;' 0000 '; FLASH 0; PAPER 0;'
                                                    '
2260 FOR V=1 TO 15
2270 FOR S=35 TO 40
2280 BEEP .01,S
2290 NEXT S
2300 NEXT V
2310 POKE 23606,0
2320 POKE 23607,60
2330 PRINT  PAPER 0; INK 9;AT 11,10*T-9;'   0000   '
2340 INK 9
2350 RETURN
2360 CLEAR 49999: LOAD 'ROTULACION'CODE 51656,768
2370 RUN
```

LOS MENSAJES DEL SPECTRUM

Cuando ejecutamos un programa en nuestro SPECTRUM, siempre aparece un mensaje, o bien de finalización, o bien de error por haber cometido una equivocación en la programación. Estos mensajes nos dicen dónde está el fallo y tenemos una pista para solucionarlo.
Todo mensaje que emite el ordenador se divide en tres partes.

0 OK, 120:2

La primera de ellas es el primer carácter que aparece (en este ejemplo el «0»), que puede ser un número o una letra, y nos sirve para encontrar en el manual la explicación de dicho mensaje. La segunda es el mensaje en sí (el «OK» del ejemplo), que nos explica lo que ha ocurrido. La última siempre son dos cifras separadas por dos puntos (:). La primera de ellas es el número de línea y la segunda es en qué comando de dicha línea se ha detenido la ejecución (en este caso en la línea 120, comando número 2).
Ahora vamos a estudiarlos uno por uno y, así podremos saber más rápidamente cómo localizar los fallos y aplicar el remedio apropiado.

El ordenador imprime siempre unos mensajes para saber las incidencias que ocurran durante la ejecución del programa.

0 OK

Nos indica que la ejecución del programa ha sido satisfactoria, sin haber encontrado la máquina ningún error.

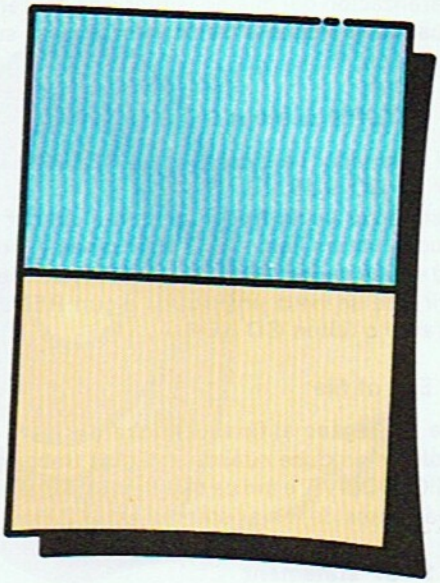
1 NEXT without FOR

Hallado **NEXT** sin existir **FOR**.
Estamos utilizando un bucle **FOR ... NEXT** y hemos olvidado escribir precisamente la sentencia **FOR ... TO**. El ordenador, sin embargo, sí ha encontrado en la ejecución del programa un **NEXT** x, sin estar previamente el **FOR** x=...TO...

2 Variable not found

El ordenador ha encontrado una variable no declarada previamente.
Es un error muy común en aquellos que antes de manejar el SPECTRUM han trabajado con otros ordenadores. Se produce cuando pretendemos trabajar con una variable sin antes haberla dado

Para evitar el mensaje **2 Variable not found** *hemos de realizar una asignación antes de utilizar la variable.*



¡!

El mensaje **R Tape loading error** aparece cuando al efectuar **LOAD** o **MERGE** no es posible cargar; o al ejecutar **VERIFY** no se ha realizado una correcta verificación.

*

Si no hemos introducido la cinta en el ZX-MICRODRIVE, o queremos trabajar con uno no existente, el ordenador imprimirá el mensaje **Microdrive not present.**

¡!

Al operar o trabajar con una variable que no tenga un valor previamente asignado, se produce el mensaje **2 Variable not found.** Se evita declarándola previamente, por ejemplo **LET x=0.**

*

Si en una sentencia **INPUT** pulsamos **STOP**, o en una **INPUT LINE** pulsamos **CAPS SHIFT+6** se produce el mensaje de aviso **H STOP in INPUT.**

un valor inicial (lo que se llama declarar una variable). Así se puede dar en **PRINT x** o **LET x=x+1** si a la variable "x" no se le ha asignado un valor previo con **LET (LET x=0), READ, INPUT, FOR** o **DIM** (en el caso que estemos trabajando con matrices y no se haya dimensionado antes con este comando).

3 Subscript wrong

Hemos asignado un subíndice erróneo en variables que los utilicen.
Interviene siempre una sentencia **DIM.** Ocurre cuando nos referimos a la tabla con un subíndice mayor que la dimensión que hicimos de ella. Así, si hemos efectuado **DIM x(90)** el error aparecerá al ejecutar: **LET x(91)=...** También aparece cuando se ha definido una matriz de dimensión cero: **DIM x(0).** Ver mensaje B.

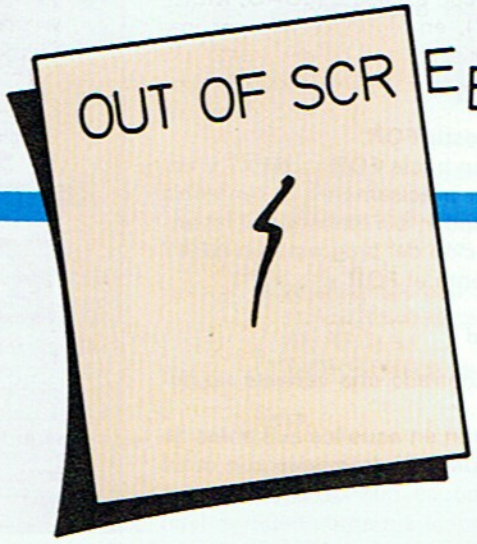
4 Out of memory

Se ha sobrepasado la capacidad de memoria de la máquina.
Nos avisa de la imposibilidad de realizar la operación pertinente al no tener suficiente capacidad el ordenador. Ocurre normalmente con **DIM,** al dimensionarla demasiado grande, con LOAD, al cargar un programa o una matriz, o MERGE, dado que la suma de los dos programas, el que se encuentra en memoria más el que se está cargando, sobrepasa la memoria libre.

5 Out of screen

Hemos pretendido imprimir texto fuera de la pantalla.
Ocurre con **INPUT** cuando llevamos escritas 23 líneas. También aparece con **PRINT AT 22...** puesto que pretendemos imprimir en una línea no existente (el SPECTRUM sólo dispone de 21 líneas de pantalla en donde podemos imprimir normalmente).

5 Out of screen *aparece cuando intentamos imprimir fuera de la pantalla.*



Cuando pretendemos rebasar la capacidad de memoria, el ordenador nos avisa con **4 Out of memory.**

6 Number too big

Ha aparecido un número demasiado grande.
Esta situación se puede dar con cualquier comando que realice alguna operación aritmética. Se sobrepasa la capacidad del ordenador para operar porque el número tiene un valor muy alto y la máquina no puede seguir. Si no existe ningún error en el programa es necesario, para evitar la paralización del mismo, situar una sentencia **IF** y cuando se llegue a cierto valor desviar su ejecución.

7 RETURN without GO SUB

El ordenador ha localizado un **RETURN** sin existir un **GO SUB.**
Este error se da frecuentemente cuando se producen varias llamadas a subrutinas y, por tanto, se sitúan varios **GO SUB** y **RETURN.** Puede ocurrir que se haya emplazado algún **RETURN** sobrante o falten **GO SUB.**

8 End of file

Se ha llegado al final del fichero.
Sólo se produce cuando estemos trabajando con MICRODRIVE, e indica que hemos pretendido leer más datos de los existentes en un fichero.

9 STOP statement

En el programa hay una sentencia **STOP**.
El ordenador, en la ejecución de un programa, se ha encontrado dicha sentencia y, por tanto, se detiene, avisándonos.

A Invalid argument

Tenemos un argumento no válido.
Ocurre sobre todo con las funciones matemáticas **ASN, ACS, LN, SQR, ^**, al intentar operar con números negativos. Esto es debido, en todos los casos menos **SQR**, a que el ordenador realiza todas las operaciones con el sistema logarítmico y éste no admite dichos números. Con **SQR** ya sabemos que no podemos operar normalmente con números negativos (es necesario recurrir a los números complejos y crearnos un programa para ello).

B Integer out of range

Tenemos un número fuera del margen permitido.
Se produce con todos los comandos, que no sean funciones matemáticas, cuyos argumentos sean números (**RUN, POKE, GO TO, LIST, DIM...**). Es debido a que dicho argumento es incorrecto para lo que se le requiere. Así con **DIM** (ver error 3) ocurre cuando se dimensiona una matriz con números negativos o mayor que 65535 (si es menor de esta cifra, 65534, aparece el mensaje 4).

C Nonsense in BASIC

No es entendido un comando.
Cuando **VAL** y **VAL$** utilizan argumentos incorrectos. Por ejemplo **VAL 10**. Si el zumbido de exceso de memoria ha sonado, independientemente del motivo, el ordenador ha podido producir «cambios extraños» en el listado del programa, siendo imposible su ejecución normal, apareciendo este mensaje. La única solución es corregirlo a mano y esperar que no haya afectado a algo más y sea necesario empezar de nuevo.

Cuando queremos seguir leyendo información en el MICRODRIVE y no hay más datos, aparece el mensaje 8 **End of file.**

Cuando por el resultado de las operaciones, una variable tome un valor demasiado alto aparecerá el mensaje 6 **Number too big.**

D BREAK-CONT repeats

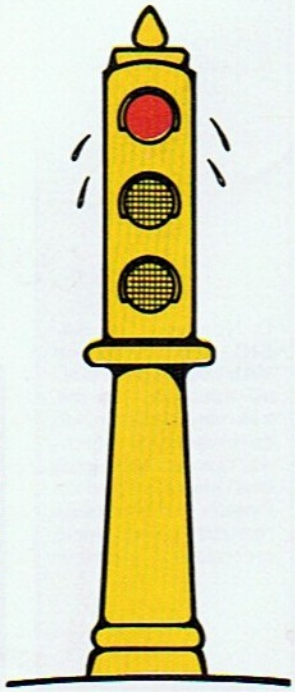
Se ha detenido voluntariamente el programa.
Cuando se pulsa BREAK (CAPS SHIFT+SPACE) en alguna operación con el casete (**LOAD, MERGE, SAVE, VERIFY**), en alguna salida por impresora o al bloquear el *scrool?* de un listado. En la primera situación, si se hace **CONTINUE** el comando se volverá a ejecutar de nuevo desde el principio. Ver el mensaje L.

E Out of DATA

Nos hemos quedado sin datos.
Hemos utilizado la sentencia **READ** más veces de lo que podemos, y tras leer todos los datos del comando **DATA** se ha continuado leyendo; al no existir más datos, se ha producido la situación errónea. Comprobando el listado, debemos contar cuantas veces se utiliza **READ** y cuantos datos hay en **DATA** para saber dónde se ha producido el error.

F Invalid file name

Al pulsar la tecla **BREAK** *provocamos la detención del programa.*



Cuando el ordenador se encuentra un STOP, *la ejecución del programa se detiene apareciendo el mensaje* 9 STOP statement.

¡!

Todo mensaje del ordenador se divide en 3 partes: la primera es un número o una letra, que nos sirve para buscar en el manual de qué mensaje se trata; la segunda es el propio mensaje y la tercera son dos cifras que nos indican el número de línea y el comando de dicha línea, respectivamente, donde se cometió el error.

El nombre del archivo (programa o *array)* es incorrecto.
Al grabar en el casete con **SAVE** se utilizó un nombre incorrecto: o con más de 10 caracteres o con ninguno.

G No room for line

No tenemos más espacio para líneas de programa.
El ordenador nos avisa, por falta de memoria, que no podemos introducir más líneas. En cualquier forma, el SPECTRUM no permite un número de orden mayor que 9999. si ejecutamos **GO TO 10000**, o similar, nos aparecerá el mensaje de error B.

H STOP in INPUT

Programa interrumpido en un **INPUT**.
Hemos provocado la detención del programa en una sentencia **INPUT** o **INPUT LINE**. En el primer caso, cuando se introduce un dato numérico, se detiene con **STOP (SYMBOL SHIFT+A)**, si es alfanumérico hay que borrar previamente la comilla de la izquierda y luego escribir **STOP**. En el segundo caso, detenemos el programa con **CAPS SHIFT+6**. En todas las situaciones aparece el mensaje H.

I FOR without NEXT

Hay un **FOR** sin el **NEXT** correspondiente.

"K INVALID COLOUR"

Si pretendemos utilizar un color no correcto, el programa se detiene apareciendo K Invalid colour.

*

C Nonsense in BASIC, además de con **VAL** y con **VAL$**, puede aparecer cuando, tras sonar el zumbido de exceso de memoria, se ha producido una alteración en el listado del programa realizada por el propio ordenador.

Cuando se nos olvida ubicar un NEXT, *al no tener bucle final, el ordenador nos avisa con el mensaje* I.

Observe que se encuentra muy relacionado con el mensaje 1. En este caso, al contrario, se nos ha olvidado situar el final del bucle con **NEXT x**, suponiendo que el comienzo es **FOR x=...** Estos errores suelen aparecer en los bucles anidados, y lo mejor es seguir fielmente el programa sobre un papel para comprobar dónde se encuentra el error.

J Invalid I/O device

No es válido el dispositivo de entrada salida.
Este mensaje se da con operaciones con el MICRODRIVE únicamente.

K Invalid colour

El color utilizado no es válido.
El parámetro de un comando gráfico, ya sea **OVER, BRIGHT, FLASH, INVERSE, INK, PA-**

PER o **BORDER**, ha sido incorrecto. No debemos olvidarnos que los tres primeros comandos sólo pueden tomar los valores 0 y 1, y el resto de 0 a 7 para los colores, 8 para caracteres transparentes y 9 para el modo de contraste (estos dos últimos números no los maneja **BORDER**).
Este mensaje es muy común también cuando estamos trabajando en BASIC y la memoria se ha visto afectada, ya sea accidental o intencionadamente.

L BREAK into Program

Programa interrumpido.
En la ejecución de un programa hemos presionado la tecla **BREAK (CAPS SHIFT+SPACE)**. Muy relacionado con D, pero aquí no se trata de operaciones con el casete.

M RAMTOP no good

RAMTOP no válido.
Al efectuar un **CLEAR x**, el parámetro x, tiene un valor demasiado pequeño o demasiado grande.

N Statement lost

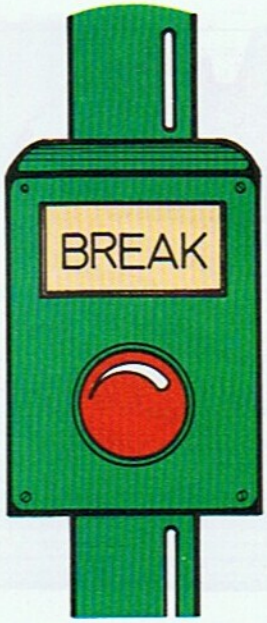
Se ha perdido una sentencia.
Ocurre cuando se produce un salto a una línea que no existe con **CONTINUE, RETURN, NEXT**. En el primer caso, se produce después de **BREAK**, y es aconsejable volver a ejecutar el programa con **GO TO...**

O Invalid Stream

Al trabajar con el MICRODRIVE, tenemos una serie no válida.

P FN without DEF

Hallado **FN** sin haber encontrado previamente **DEF FN**.
Cuando trabajamos con una función es necesario que vayan emparejados dos comandos: **DEF FN** y **FN**. Este mensaje de error indica que el ordenador no ha encontrado la definición de la función cuando se le ha pedido trabajar con ella.



Si aparece D BREAK-cont repeats *o* L BREAK into Program *es que hemos detenido voluntariamente el programa apretando* BREAK.

Q Parameter error

Hemos introducido un parámetro erróneo.
Ocurre con el comando **FN**. Al operar se ha encontrado con un parámetro no correcto; por ejemplo, una cadena alfanumérica en el lugar de numérica, o si en nuestra función hay potenciaciones, ha aparecido un número negativo al operar.

R Tape loading error

Error de carga en el casete.
Cuando hemos procedido a cargar en el ordenador, con **LOAD** o **MERGE**, o a verificar la graba-

El ordenador no «sabe» operar siempre con números negativos.

$$(-2)^2$$

¡!

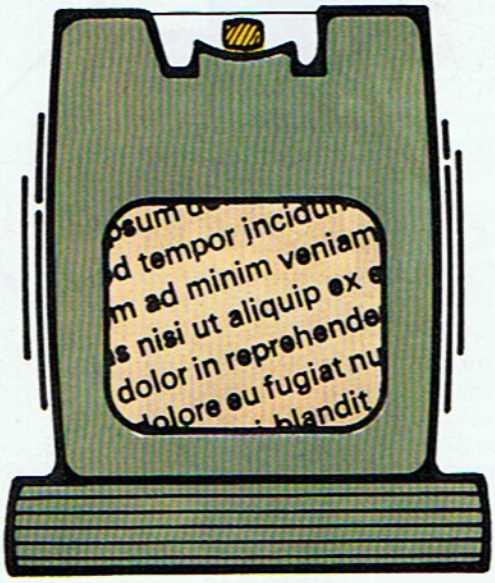
Cuando leemos **(READ)** más datos de lo que se encuentra en **DATA** aparece el mensaje **E Out of DATA**.

*

Cuando los argumentos de algunas funciones son incorrectos aparece el mensaje **A Invalid argument.**

*

Si pretendemos acceder a un dato en una tabla con un subíndice incorrecto (negativo o más alto que la dimensión de la propia tabla) aparece el mensaje **3 Subscript wrong.**



MICRODRIVE FULL

Cuando operamos con un cartucho de MICRODRIVE en el que no cabe más información aparece el mensaje Microdrive full.

MENSAJES CON EL ZX-MICRODRIVE

Microdrive not present

No hay MICRODRIVE.
Este mensaje nos puede aparecer en dos situaciones. La primera es cuando pretendemos trabajar sin tener la cinta del MICRODRIVE introducida en el aparato. Si hemos indicado que queremos trabajar con un ZX MICRODRIVE que no tenemos conectado (por ejemplo, tenemos un solo aparato y señalamos que queremos trabajar con el segundo), también aparece este mensaje.

Microdrive full

El MICRODRIVE está completo.
No podemos cargar más información en la cinta de MICRODRIVE al no disponer de espacio. Podemos saber continuamente el espacio libre si realizamos de una manera periódica **CAT:** aparece un listado de los programas y ficheros en la cinta y los Kbytes libres.

File not found

Fichero no hallado.
Al ejecutar **LOAD** o **MERGE** el programa indicado no ha sido encontrado. Realizando **CAT** sabremos si de verdad falta o hemos cometido un error al escribirlo.

Verification has failed

Verificación fallada.
Se ha producido un error al verificar **(VERIFY)** un programa, posiblemente por un fallo en la grabación **(SAVE).**

ción **(VERIFY)** ha ocurrido un error que hace imposible dicha carga. En la primera situación hay que repetir la operación, para comprobar si vuelve a aparecer el mensaje. Si se estaba ejecutando **VERIFY** se debe volver a grabar con **SAVE.** Este tipo de error puede surgir por muy diferentes motivos: desde que el aparato de grabación se ha detenido accidentalmente, hasta un fallo en él o en la cinta. En este caso, se puede saber mirando si está doblada en algún punto, o si mirando el cuentavueltas del aparato, el mensaje R aparece siempre en el mismo número; se impone una sustitución de ésta.

¡!

Cuando realizamos varias llamadas a subrutinas es posible que aparezca el mensaje **7 RETURN without GO SUB** al colocar más comandos **RETURN** que **GO SUB.**

*

Con **0 OK** el ordenador nos indica que el programa se ha ejecutado correctamente hasta el final.

*

Cuando ejecutamos un programa en nuestro SPECTRUM siempre aparece un mensaje, o bien de fin, o bien de error, por haber cometido una equivocación en la programación.

C Nonsense in BASIC *aparece cuando un comando no es entendido por el ordenador.*

MANIPULANDO BITS

Hasta el presente capítulo, todas las instrucciones discutidas mantenían el objetivo común de efectuar una determinada operación sobre un octeto o grupo de ellos, pero no podíamos actuar sobre un bit concreto del cual, por cualquier razón, fuera preciso escrutar o modificar su contenido. Precisamente, una de las mejoras que el Z 80 incorpora frente a su predecesor el 8080, la constituye el numeroso grupo de instrucciones encargadas de gestionar información a nivel de bit individual (312 nuevas instrucciones).

TEST, SET Y RESET

En la página 649 bajo la denominación «MANIPULACION DE BITS» están recogidas todas las instrucciones encargadas de examinar (TEST) el contenido del bit señalado dentro de un registro o posición de memoria determinada, colocarlo a 1 (SET) fuera cual fuera su estado, o fijarlo a 0 (RESET).

Los mnemónicos asociados con estas operaciones son BIT b,s SET b,s y RES b,s respectivamente, donde s es el contenido del registro o posición de memoria (8 bits) a la que acceder y b, el lugar ocupado por el bit individual, sobre el cual vamos a actuar.

Recordemos el convenio seguido en toda la literatura informática, según el cual dentro de un octeto, los bits que lo constituyen van numerados de 0 a 7 comenzando por el de la derecha.

Supongamos, por ejemplo, que el acumulador contiene en el momento previo a efectuar una de estas instrucciones 10110011. Comprobemos el efecto que sobre el valor almacenado en A obran la ejecución sucesiva de algunas de estas órdenes:

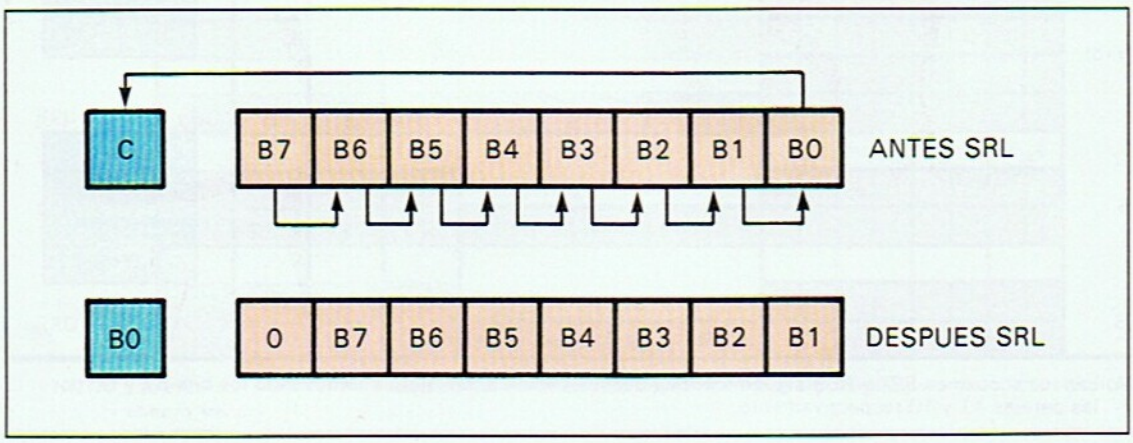
INSTRUCCION	ACUMULADOR
SET 2,A	10110111
SET 4,A	10110111
RES 7,A	00110111
RES 6,A	00110111

Fácilmente se comprueba que tanto las instrucciones SET como las de RESET son independientes del valor original del bit afectado, es decir, fijan su contenido a 1 (SET) o a 0 (RES) sin atender a su estado anterior.

Otra circunstancia a tener en cuenta es que implementarlas, no provoca cambio en el registro de indicadores F. Sin embargo, diferente es lo que ocurre cuando nuestro objetivo es escrutar (TEST) un bit específico.

Supongamos ahora, que el acumulador mantiene el valor obtenido tras la instrucción RES 6,A. Efectuemos algunas operaciones de TEST sobre algunos bits del byte almacenado en A:

INSTRUCCION	ACUMULADOR	INDICADOR DE CERO
BIT 0,A	00110111	0
BIT 6,A	00110111	1



¡!

Todas las instrucciones de rotación y desplazamiento afectan a los indicadores S, Z, P/V y C, a excepción de las cuatro anteriores.

*

Al ejecutar RLD o RRD rotamos los dígitos decimales de un número expresado en formato BCD.

*

Las instrucciones de desplazamiento tienen especial utilidad en la construcción de rutinas para la multiplicación y división multibyte.

La primera consecuencia que extraemos es que tras una instrucción BIT, el contenido del registro o posición de memoria implicada no varía. En nuestro ejemplo, el acumulador no ha experimentado cambio alguno.

La segunda es que el resultado se pone de manifiesto en función de la bandera de cero del registro F, es decir, si el bit es 0, el indicador Z contendrá 1, y 0 en caso contrario.

Estas circunstancias son, en particular, muy útiles cuando tratemos de determinar el valor de un bit concreto sin necesidad de modificar el contenido de A. Por ejemplo, las dos parejas de instrucciones siguientes conducen al mismo objetivo: determinar si el bit B6 del byte almacenado en el registro B es 1.

```
LD A,B          LD A,B
AND 40h         BIT 6,A
```

La primera secuencia, se vale del byte 40 como máscara para detectar si efectivamente el bit considerado es 1, modificándose para ello el contenido de A. La segunda realiza el mismo trabajo, pero A no se ve alterado. Ambas precisan 5 bytes para ser implementadas, y según las circunstancias nos decidiremos por una u otra técnica.

```
CONVERSOR BCD-ASCII

23296           LD A,02          ;pantalla superior
23298           CALL CHAN-OPEN   ;abre canal 2
23301           LD HL,9C40       ;origen de los números BCD
23304           LD C,cantidad    ;¿cuántos van a ser convertidos?
23306   BCD     LD B,02          ;cada número BCD tiene dos
                                 ;dígitos
23308   LOOP    PUSH BC          ;guarda BC
23309           PUSH HL          ;guarda HL
23310           XOR A            ;borra el acumulador
23311           RLD              ;rotación decimal a la izquierda
23313           ADD A,30         ;sitúa en el nibble de mayor
                                 ;orden 3
23315           RST 10           ;imprime carácter ASCII
23316           POP HL           ;recupera HL
23317           POP BC           ;recupera BC
23318           DJNZ LOOP        ;efectúa el bucle dos veces (dos
                                 ;dígitos)
23320           INC HL           ;toma el siguiente BCD
23321           DEC C            ;queda uno memos que procesar
23322           JR NZ BCD        ;¿queda alguno por procesar?
23324           RET              ;vuelve al BASIC.
```

ROTACIONES

En la página 653 están recogidas todas las instrucciones de rotación y desplazamiento. Centrándonos en las de rotación, las cuatro primeras

MNEMONICO	CODIGO MAQUINA	REGISTRO F 7 (S)	6 (Z)	5	4 (H)	3	2 (P/V)	1 (N)	0 (C)	Nº BYTES	CICLOS MAQ.	CICLOS RELOJ	NOTAS
BIT b,r	1 1 0 0 1 0 1 1												
	0 1 x b x x r x	x	↕		1		x	0	•	2	2	8	
BIT b,(IX+d)	1 1 0 1 1 1 0 1												
	1 1 0 0 1 0 1 1												
	x d x												
	0 1 x b x 1 1 0	x	↕		1		x	0	•	4	5	20	
BIT b,(IY+d)	1 1 1 1 1 1 0 1												
	1 1 0 0 1 0 1 1												
	x d x												
	0 1 x b x 1 1 0	x	↕		1		x	0	•	4	5	20	
SET b,S	1 1	•	•	•	•	•	•	•	•				
RES b,S	1 0	•	•	•	•	•	•	•	•				

	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111
(HL)	110

S
r
(IX+d)
(IY+d)

NOTA: Las instrucciones SET y RES siguen idéntica decodificación a BIT, pero sustituyendo los bits B7 y B6 por las parejas 11 y 10 respectivamente.

líneas contienen los códigos asociados con estas operaciones.

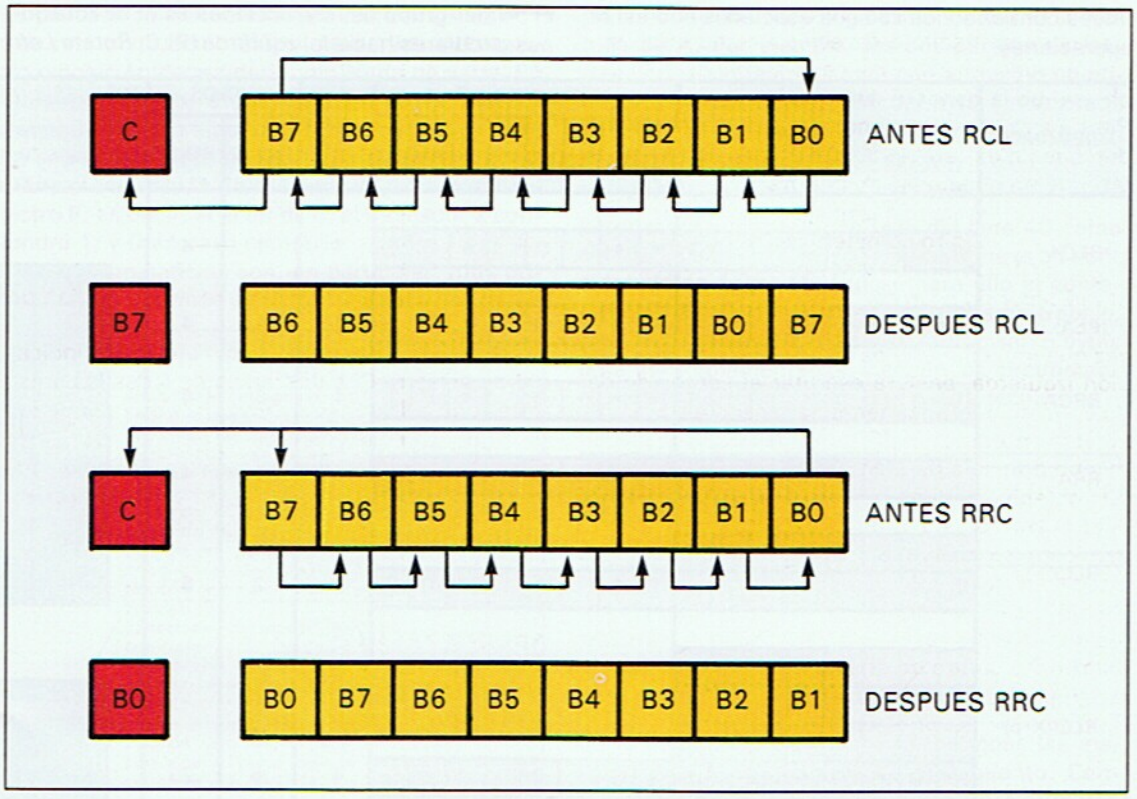
El primer grupo de instrucciones es el de rotaciones circulares hacia la izquierda (RLC, *Rotate Left*

MNEMONICO	CODIGO MAQUINA	REGISTRO F 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Nº BYTES	CICLOS MAQ.	CICLOS RELOJ	NOTAS
RLCA	0 0 0 0 0 1 1 1	•	•		0		•	0	↕	1	1	4	
RLA	0 0 0 1 0 1 1 1	•	•		0		•	0	↕	1	1	4	
RRCA	0 0 0 0 1 1 1 1	•	•		0		•	0	↕	1	1	4	
RRA	0 0 0 1 1 1 1 1	•	•		0		•	0	↕	1	1	4	
RLCr	1 1 0 0 1 0 1 1 0 0 0 0 0 x r x	↕	↕		0		P	0	↕	2	2	8	
RLC(IX+d)	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 x d x 0 0 0 0 0 1 1 0	↕	↕		0		P	0	↕	4	6	23	
RLC(IY+d)	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 x d x 0 0 0 0 0 1 1 0	↕	↕		0		P	0	↕	4	6	23	
RL s	0 1 0	↕	↕		0		P	0	↕				
RRC s	0 0 1	↕	↕		0		P	0	↕				
RR s	0 1 1	↕	↕		0		P	0	↕				
SLA s	1 0 0	↕	↕		0		P	0	↕				
SRA s	1 0 1	↕	↕		0		P	0	↕				
SRL s	1 1 1	↕	↕		0		P	0	↕				
RLD	1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 1	↕	↕		0		P	0	•	2	5	18	
RRD	1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 1	↕	↕		0		P	0	•	2	5	18	

	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111
(HL)	110

S
r
(IX+d)
(IY+d)

NOTA: Las instrucciones donde en su código máquina sólo se dan tres bits siguen idéntica decodificación que los anteriores.



¡!

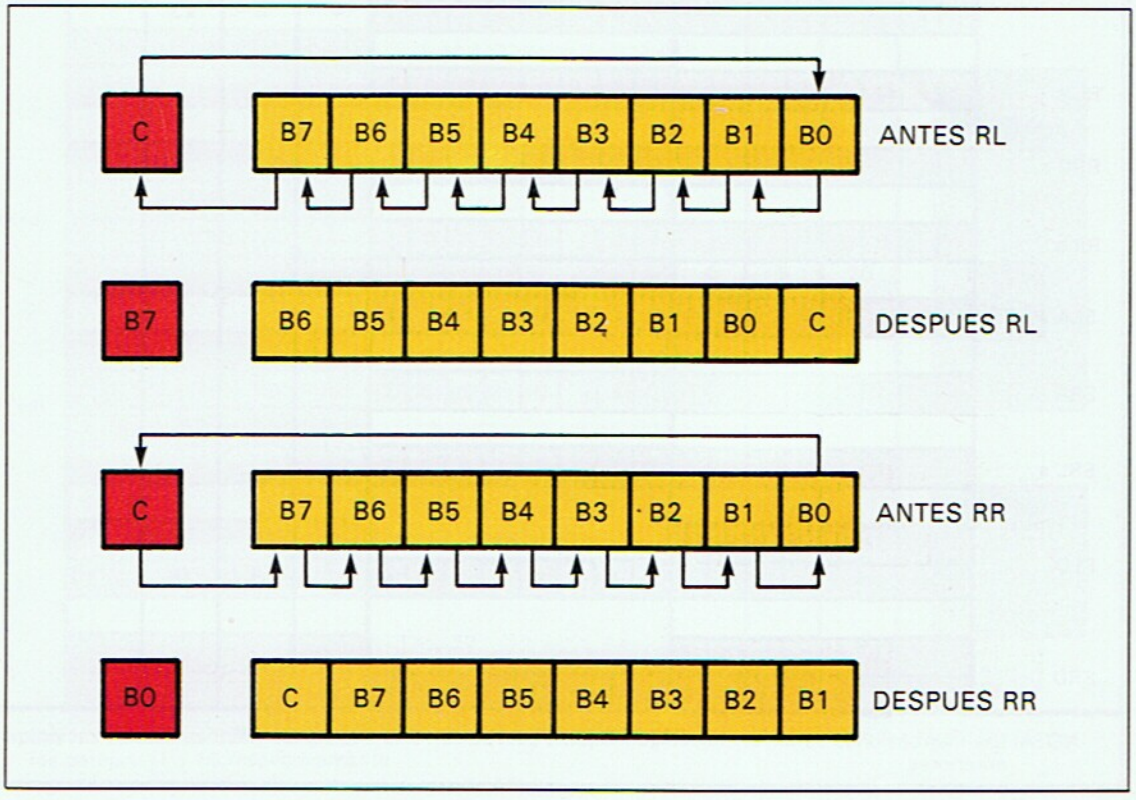
La instrucción de rotación derecha RR permite mediante aplicaciones sucesivas a un determinado octeto seleccionar el acceso a una secuencia de subrutinas, previamente escogida por nosotros.

*

Los grupos de manipulación de bit, rotación y desplazamiento constituyen un nuevo bloque de instrucciones, solamente contemplado en el Z 80, no así en su predecesor el 8080.

Circular). Su mecanismo queda descrito en el diagrama de la figura, es decir, todos los bits del octeto considerado giran una posición hacia la izquierda, y el contenido del último se duplica en el lugar dejado vacante por primero y en el indicador de arrastre C.

La rotación derecha circular (RRC, *Rotate Right Circular)* es, en esencia, idéntica a la anterior,

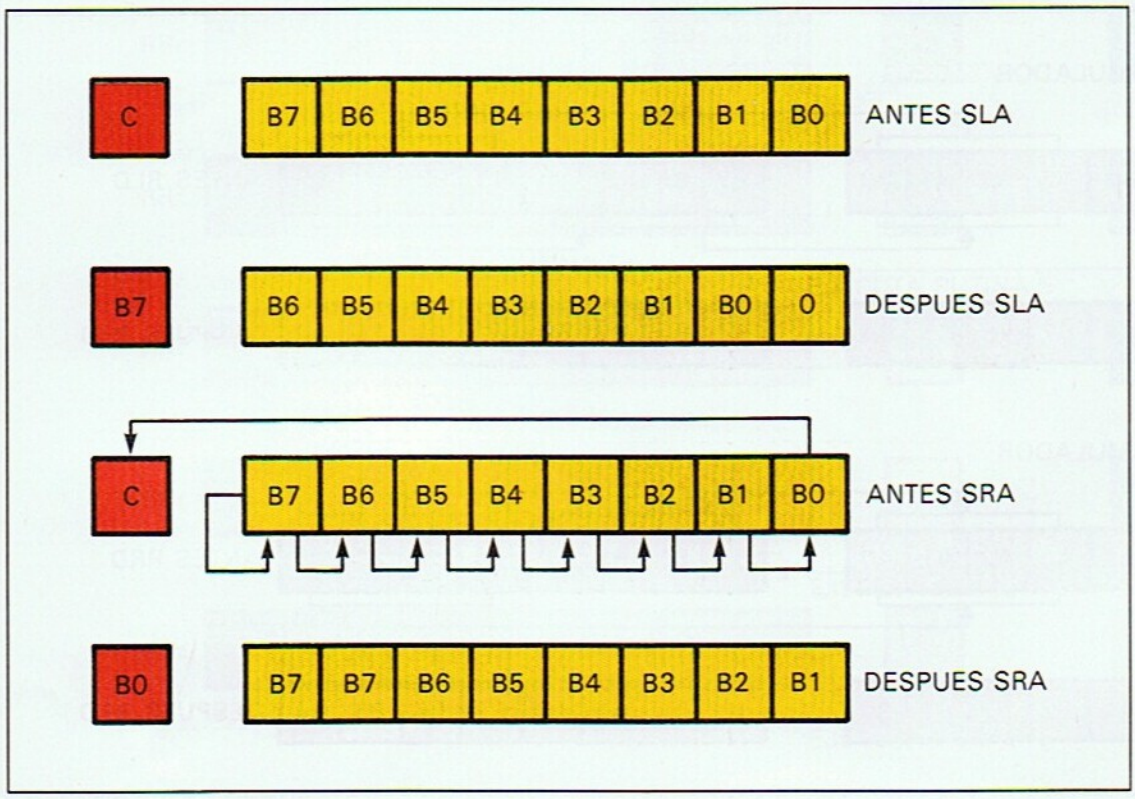


pero ahora el giro se produce en sentido contrario, es decir, hacia la derecha y como el bit sobrante sería el B0, éste pasa a ocupar el lugar B7 y el de la bandera de arrastre.
Para averiguar, por ejemplo, el bit de mayor orden que no es cero dentro de un determinado octeto, se emplean con mucha frecuencia las instrucciones de rotación izquierda (RL, *Rotate Left).* Se trata de un giro completo de los bits del octeto especificado al que añadimos un noveno bit: el indicador de acarreo.
Puesto que B7 es transferido al indicador C cada vez que implementamos una instrucción de rotación izquierda, bastará escrutar el contenido de éste, hasta que finalmente almacene un 1.
La rotación derecha (RR, *Rotate Right)* sigue el mismo principio de efectuar el giro de los nueve bits obtenidos al añadir el bit C al octeto sobre el cual operamos, aunque por supuesto, el desplazamiento se realiza en sentido contrario a las agujas del reloj.
Su aplicación resulta inmediata en problemas en los que sólo se necesite acceder a determinadas subrutinas dentro de un bloque de éstas, y en un orden prefijado, de tal forma que no siempre se ejecuten todas, sino tan sólo las que necesitemos.
En el diagrama hemos representado un ejemplo de como se implementaría esta técnica manejando una instrucción de rotación derecha. Observaremos que es posible fijar 256 combinaciones diferentes según nuestras necesidades.

Al final de la tabla de la página 653 se encuentran cuatro instrucciones redundantes con las anteriores y que tan sólo actúan sobre el contenido del acumulador. Se trata de RLCA, RRCA, RLA y RRA.
Aun a pesar de poder resultar repetitivas, los fabricantes del Z 80 prefirieron mantenerlas para conservar la compatibilidad con el microprocesador 8080 y bajo determinadas condiciones nosotros podemos emplearlas con cierta ventaja sobre sus equivalentes, pues tan sólo precisan de un byte para ser implementadas.
Estas cuatro instrucciones sólo afectan al indicador de arrastre C, a diferencia de todas las anteriores que además modifican el contenido de los de signo, paridad/sobrepasamiento, y cero. Por tanto, cuando tras una operación de rotación solamente sea necesario tener presente el acarreo conseguiremos ahorrar tiempo de proceso y espacio de memoria implementando estas últimas.

DESPLAZAMIENTOS

Las instrucciones de desplazamiento constituyen un arma de inestimable valor cuando se trate de



¡!

Las instrucciones RLCA, RRCA, RLA y RRA solamente pueden trabajar con el contenido del acumulador y además, afectan exclusivamente al indicador de arrastre C.

*

La instrucción TEST escruta un bit específico dentro de un octeto y revela su estado activando o no la bandera de cero Z.

*

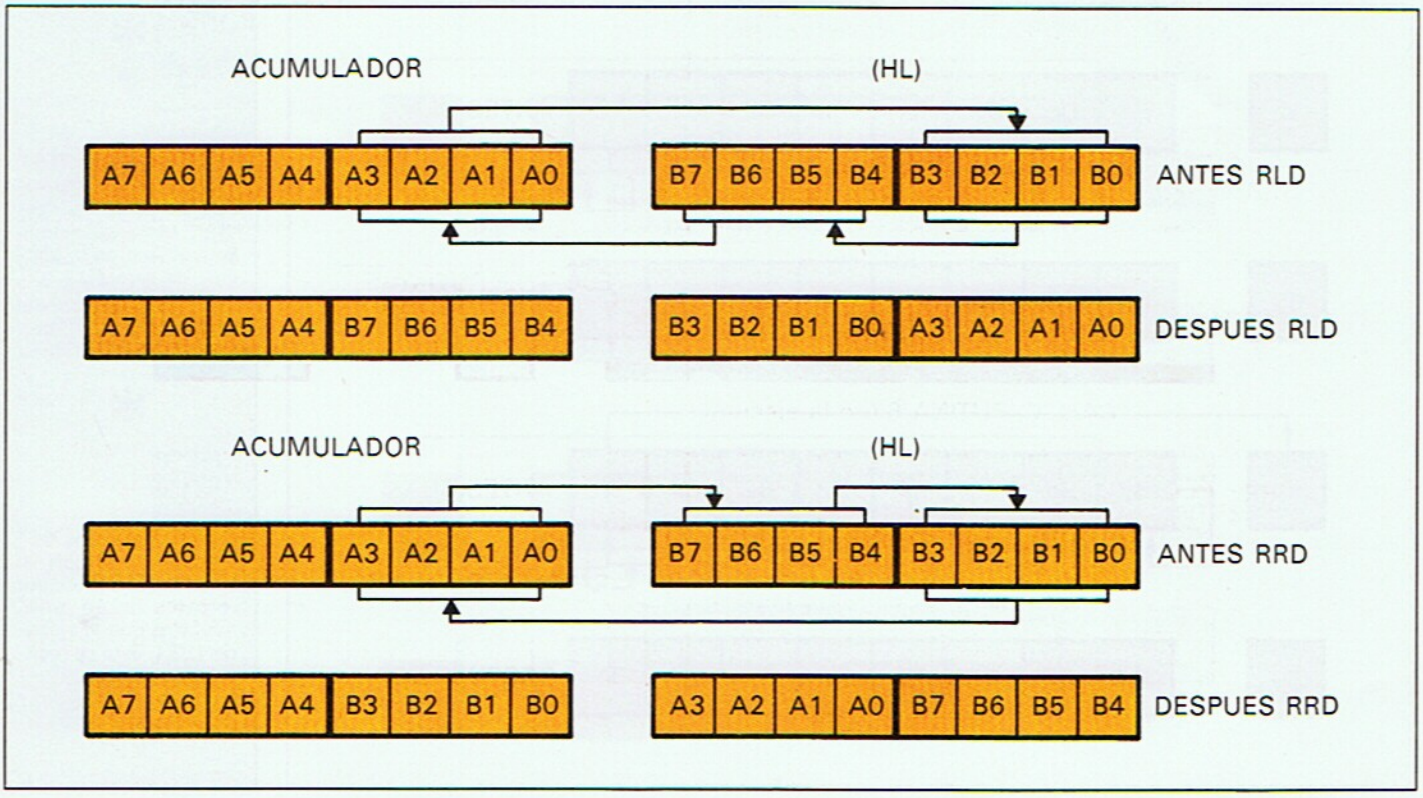
Las instrucciones SET y RES modifican el contenido del octeto implicado en ella, mientras que al ejecutar TEST, éste se mantiene inalterado.

efectuar rutinas que resuelvan las operaciones de multiplicación y división de varios bytes.
En su mecanismo interviene de manera protagonista el indicador de arrastre C, aunque tras procesarlas, las banderas Z, S y P/V serán afectadas según el resultado obtenido, al igual que ocurría con las instrucciones de rotación.
En los gráficos correspondientes se representa de forma esquemática su funcionamiento. Observemos que la instrucción de desplazamiento aritmético a la izquierda SLA *(Shift Left Arithmetic),* provoca que el valor almacenado en el octeto afectado sea multiplicado por dos, proporcionando una señal de sobrepasamiento (se activa el indicador de arrastre), cuando el resultado sea mayor o igual de 256.
Por el contrario, la instrucción SRA *(Shift Right Arithmetic,* desplazamiento aritmético a la derecha) divide el contenido del registro dado entre dos, situándose el resto de esta división (el microprocesador supone que los números están expresados en complemento a dos) en el indicador C.
Finalmente, SRL *(Shift Right Logic,* desplazamiento lógico hacia la derecha), efectúa la división por dos del valor almacenado en el registro implicado por la instrucción, pero ahora considerando que está expresado en formato binario.
Resulta útil considerar el uso conjunto de las instrucciones de rotación y las de desplazamiento al objeto de conseguir trasladar a la izquierda o derecha un grupo de varios bytes. Por ejemplo, la siguiente secuencia desplaza a la izquierda bit a bit el contenido del par HL, colocando en el lugar vacante un 0, lo cual equivale a la multiplicación por 2 del valor almacenado en dicho par:

```
SLA L
RL H
```

ROTACIONES EN BCD

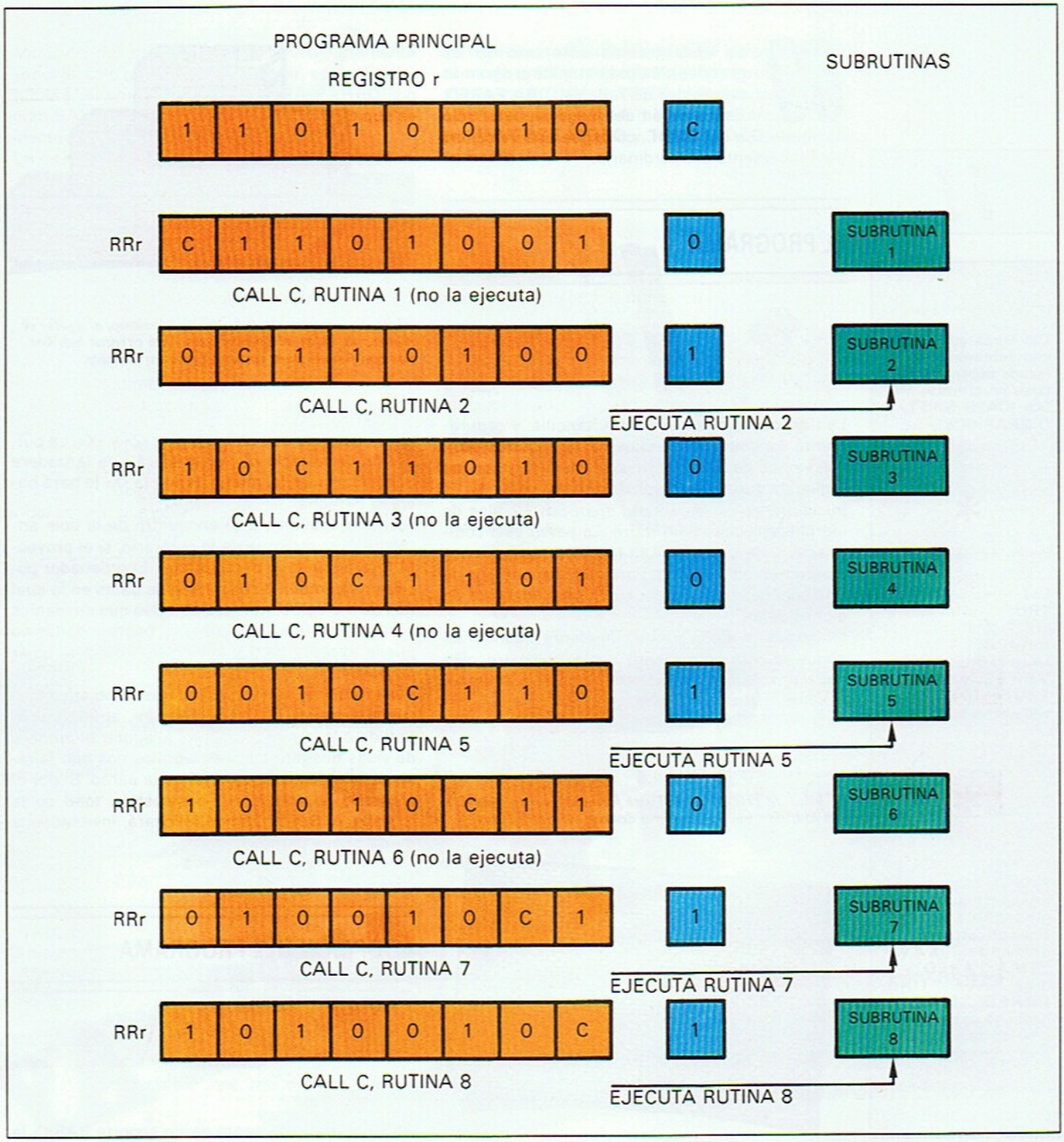
Para terminar con el grupo de instrucciones de rotación y desplazamiento nos queda comentar dos de ellas, las cuales sirviéndose de direccionamiento indirecto (efectúan su trabajo sobre el octeto señalado por el par HL), ejecutan rotaciones a la izquierda o derecha de grupos de 4 bits (nibbles).
Al igual que antes, para no perder la información sobrante al ejecutar el giro (rotábamos bit a bit y el saliente pasaba al indicador C), el nibble saliente, ahora se recoge en los 4 bits de menor peso del acumulador A, mientras que los previamente contenidos allí quedan almacenados en el nibble vacío del octeto considerado. En la figura se representa el proceso seguido.



Por estas razones, las instrucciones RLD *(Rotate Left Decimal,* rotación decimal hacia la izquierda) y RRD *(Rotate Right Decimal,* rotación decimal hacia la derecha) son de especial utilidad cuando estemos manejando números expresados en formato binario codificado en decimal (BCD), pues efectúan una rotación entre los dígitos decimales almacenados en la posición de memoria indicada por HL y el acumulador.

En el próximo capítulo serán tratadas las instrucciones de entrada/salida de datos entre el microprocesador y periféricos.

Asimismo, abordaremos un tema de especial interés: las interrupciones.

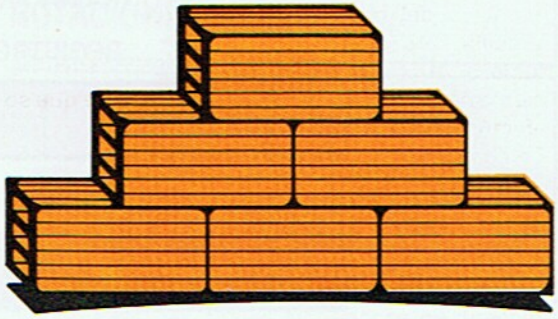


TIRA PARED

Nos encontramos ante uno de los grandes clásicos entre los programas de rapidez de reflejos: TIRA PARED. Esta versión del juego, aunque sólo haciendo uso del BASIC, consigue una velocidad auténticamente extraordinaria.

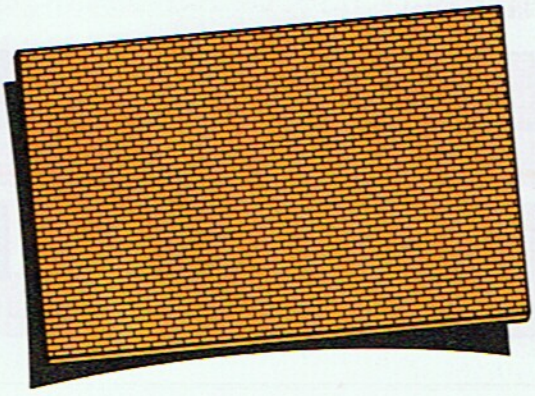
EL PROGRAMA

La finalidad del juego es muy sencilla, y seguramente conocida por la mayoría de nosotros. Aun así, vamos a recordar brevemente cuales son las reglas de nuestro programa TIRA PARED.
Inicialmente en la pantalla aparecen 12 filas de ladrillos simulando un muro. La pared está compuesta por un total de 198 ladrillos que debemos destruir en su totalidad. Para ello, disponemos de una bola desenladrilladora y de una lanzadera de dichas bolas.
La lanzadera actúa a modo de «cama elástica» en donde la bola botará para iniciar la destrucción del muro. Nosotros podemos controlar la lanzadera haciendo uso de las teclas «Z» y «X». La pulsación de la tecla «Z», desplazará a la lanzadera hacia la izquierda, mientras que la «X» lo hará hacia la derecha.
Siempre debemos ir al encuentro de la bola antiladrillo, puesto que de lo contrario, si el proyectil toca el suelo lo perderíamos. El ordenador posee una pantalla de captación de datos en la cual nos da a elegir el número de vidas que deseamos para una partida. El máximo de oportunidades no puede exceder de 9 y el mínimo no ha de bajar de 3.
Por último, si no logramos nuestro objetivo (derribar el muro), nuestro Spectrum, al finalizar la partida o lo que es lo mismo, al agotar el número de vidas nos dirá cuantos ladrillos nos han faltado para la destrucción total de la pared. Si por el contrario, logramos esa destrucción total de la muralla, el Spectrum nos felicitará, invitándonos a una nueva partida.



Al finalizar la partida, o lo que es lo mismo, al agotar el número de vidas, el Spectrum nos dirá cuántos ladrillos nos han faltado para conseguir nuestro objetivo.

El muro de TIRA PARED está formado por un total de 198 ladrillos, que es necesario destruir en su totalidad.



ADOPCION DEL PROGRAMA

Al constar simplemente de un soporte BASIC, la

¡!

Las letras que aparecen subrayadas en el listado deben introducirse en el modo gráfico (**CAPS SHIFT+9** O **GRAPHICS**).

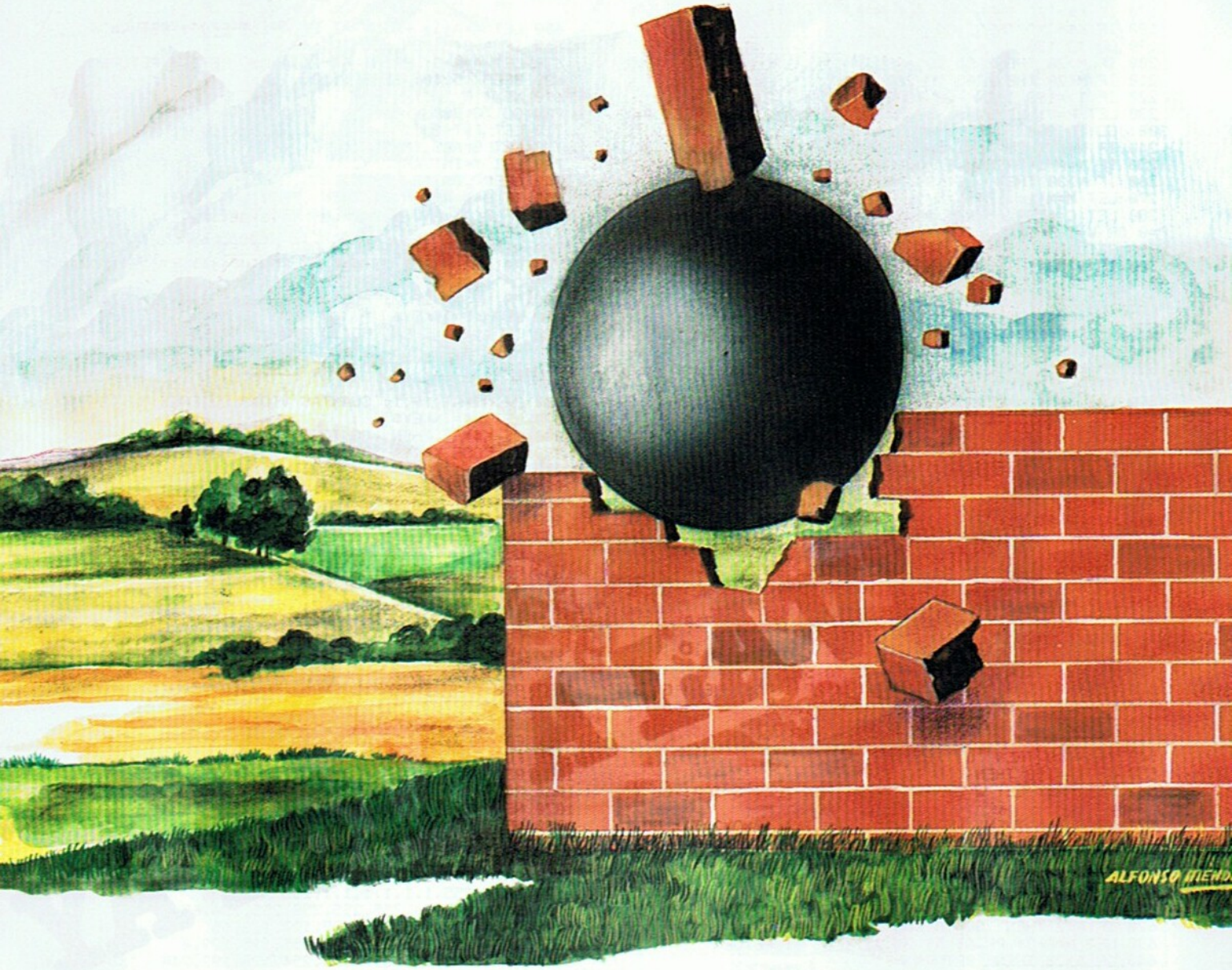
*

Los literales **INV.** y **TRUE**, subrayados y entre corchetes, corresponden a los caracteres de control **INVERSE VIDEO** y **TRUE VIDEO**, respectivamente.

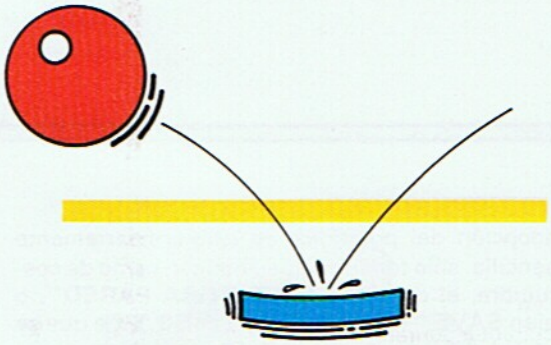
*

Para la adopción del programa sólo tenemos que emplear, como de costumbre, el comando **SAVE "TIRA PARED"**, o bien **SAVE "TIRA PARED" LINE 1**, para que se efectúe el proceso de autoejecución.

adopción del programa es extraordinariamente sencilla; sólo tenemos que emplear, como de costumbre, el comando **SAVE "TIRA PARED"**, o bien **SAVE "TIRA PARED" LINE 1**, para que se efectúe el proceso de autoejecución.
En cuanto a las letras que aparecen subrayadas, ya habremos adivinado que debe introducirse en el modo gráfico, y al igual que en otros listados, los literales **INV.** y **TRUE**, subrayadas y entre corchetes, corresponden a los caracteres de control **INVERSE VIDEO** y **TRUE VIDEO**, respectivamente.



```
   1 REM ************************
   2 REM * J.M.MAYORAL SERRANO *
   3 REM ************************
   4 REM * TIRA PARED (C) 1985 *
   5 REM ************************
   8 POKE 23658,8
  10 GO SUB 840
  15 IF NOT PEEK 32767 THEN  GO SUB 1000
  20 LET TT=0: LET T=0: LET P=1: BORDER 0: INK 0: PAP
ER 7: CLS
  30 FOR M=1 TO 12 STEP 2
  40 FOR N=0 TO 30 STEP 2
  50 PRINT AT M+3,N; INK 7; PAPER 2;"AB";AT M+4,N;"BA

  60 NEXT N: NEXT M
  70 LET U=0: LET V=0
  80 LET A=14: LET T=0: LET W=0
  90 FOR R=1 TO TOPE
 100 RANDOMIZE : LET M=15: LET N=INT (RND*3)+13
 110 LET G=530: LET P=0: LET A=13
 120 PRINT AT 21,0;"          EFG
 130 GO SUB G: PRINT AT U,V;" "
 140 PRINT  INK 4;AT M,N;"C": LET U=M: LET V=N
 150 IF M=20 THEN  PRINT  INK 4;AT M,N;"C"
 160 LET K$=INKEY$
 170 IF K$="Z" THEN  GO SUB 620
 180 IF K$="X" THEN  GO SUB 660
 190 GO TO 130
 200 IF M>20 THEN  GO TO 680
 210 IF M<20 THEN  GO TO 260
 220 IF T>=198 THEN  GO TO 930
 230 LET P=0: LET W=0: IF N=A+1 OR N=A+2 THEN  LET G=
300: GO TO 300
 240 IF N=A THEN  LET G=500: GO TO 510
 250 IF N=A-1 THEN  LET G=350: GO TO 350
 260 IF N>30 THEN  LET G=400: GO TO 400
 270 LET M=M+1: LET N=N+1
 280 LET C=ATTR (M,N): IF C<>56 THEN  GO SUB 710: IF
P=0 OR W=1 THEN  LET G=300
 290 RETURN
 300 IF N>30 THEN  LET G=350: GO TO 350
 310 IF M<1 THEN  LET W=1: LET G=200: GO TO 200
 320 LET M=M-1: LET N=N+1
 330 LET C=ATTR (M,N): IF C<>56 THEN  GO SUB 710: LET
P=1: LET G=200+150*W
 340 RETURN
 350 IF M<1 THEN  LET W=1: LET G=400: GO TO 400
 360 IF N<1 THEN  LET G=300: GO TO 300
 370 LET M=M-1: LET N=N-1
 380 LET C=ATTR (M,N): IF C<>56 THEN  GO SUB 710: LET
P=1: LET G=400-100*W
 390 RETURN
 400 IF M>20 THEN  GO TO 680
 410 IF M<20 THEN  GO TO 460
 420 IF T>=198 THEN  GO TO 930
 430 LET P=0: LET W=0: IF N=A+3 THEN  LET G=300: GO T
O 300
 440 IF N=A+2 THEN  LET G=500: GO TO 500
 450 IF N=A OR N=A+1 THEN  LET G=350: GO TO 350
 460 IF N<1 THEN  LET G=200: GO TO 200
 470 LET M=M+1: LET N=N-1
 480 LET C=ATTR (M,N): IF C<>56 THEN  GO SUB 710: IF
P=0 OR W=1 THEN  LET G=350
 490 RETURN
 500 IF M<1 THEN  LET G=530: GO TO 590
 510 LET M=M-1: LET C=ATTR (M,N): IF C<>56 THEN  GO S
UB 710: LET G=530
 520 RETURN
 530 IF M>20 THEN  GO TO 680
 540 IF M<20 THEN  GO TO 590
 550 IF T>=198 THEN  GO TO 930
 560 LET P=0: LET W=0: IF N=A+2 THEN  LET G=300: GO T
O 300
 570 IF N=A+1 THEN  LET G=300+(50*(INT (RND*2))): GO
TO G
 580 IF N=A THEN  LET G=350: GO TO 350
 590 LET M=M+1: RETURN
 600 IF A<1 THEN  RETURN
 610 LET A=A-1: PRINT AT 21,A;"EFG ": RETURN
 620 IF A<2 THEN  GO TO 600
 630 LET A=A-2: PRINT AT 21,A;"EFG  ": RETURN
 640 IF A>28 THEN  RETURN
```



La lanzadera actúa a modo de cama elástica, en la cual la bola rebotará para iniciar la destrucción del muro.

```
 650 LET A=A+1: PRINT AT 21,A-1;" EFG": RETURN
 660 IF A>27 THEN  GO TO 640
 670 LET A=A+2: PRINT AT 21,A-2;"  EFG": RETURN
 680 BEEP .5,40: BEEP 1,10
 690 NEXT R
 700 GO TO 780
 710 LET T=T+1
 720 LET B=ABS (M-N)
 730 LET Y=INT (B/2)*2
 740 BEEP .01,40
 750 IF B=Y AND N<31 THEN  PRINT AT M,N+1;" "
 760 IF B<>Y AND N>0 THEN  PRINT AT M,N-1;" "
 770 RETURN
 780 REM [INV.] FIN DE JUEGO [TRUE]
 790 PRINT  PAPER 2; INK 6; FLASH 1;AT 17,2;"TE HAN F
ALTADO ";198-T;" LADRILLOS"
 800 PRINT AT 19,2;"PULSA UNA TECLA PARA EMPEZAR"
 810 LET K$=INKEY$
 820 IF K$="" THEN  GO TO 810
 830 RUN
 840 REM [INV.] INTRO TOPE [TRUE]
 850 CLS
 860 PRINT AT 19,1;"CUANTAS VIDAS QUIERES ?  (3-9)"
 870 LET K$=INKEY$
 880 IF K$="" THEN  GO TO 870
 890 IF CODE K$<51 OR CODE K$>57 THEN  GO TO 870
 900 BEEP .1,40
 910 LET TOPE=VAL K$
 920 RETURN
 930 REM [INV.] PARED DESTRUIDA [TRUE]
 940 INK 6: PAPER 1: BORDER 1
 945 CLS
 950 PRINT  FLASH 1;AT 10,3;"  ERES UN TIO MUY HABIL!
 "
 960 PRINT  OVER 1;AT 13,2;"PULSA UNA TECLA PARA EMPE
ZAR"
 970 LET K$=INKEY$
 980 IF K$="" THEN  GO TO 960
 990 RUN
1000 REM [INV.] GRAFICOS DE USUARIO [TRUE]
1010 POKE 32767,1
1020 LET A$="ABCEFG"
1030 FOR N=1 TO LEN A$
1040 FOR F=0 TO 7
1050 READ A
1060 POKE USR A$(N)+F,A
1070 NEXT F
1080 NEXT N
1090 RETURN
1100 REM [INV.] DATAS [TRUE]
1110 DATA 255,128,128,128,128,128,128,255
1120 DATA 255,1,1,1,1,1,1,255
1130 DATA 60,126,255,255,255,255,126,60
1140 DATA 127,255,255,255,127,1,3,15
1150 DATA 255,255,255,255,255,195,129,0
1160 DATA 254,255,255,255,254,128,192,240
```

SPECTRUM 128: EL NUEVO SINCLAIR

Cuando en el año 1983 los aficionados a la informática recibíamos la noticia de la aparición en el mercado de un nuevo microordenador a un precio revolucionario comparado con los manejados por aquel entonces, un cierto rictus de susceptibilidad se apoderó de todos nosotros.

Se trataba del Sinclair ZX Spectrum. ¿Pero quién era Sinclair? Sobre esta marca las únicas noticias que teníamos conducían directamente al nanoordenador ZX 81, un modesto equipo con teclado a base de membranas sensibles y con 1 Kbyte de memoria en su configuración base.

Pero el nuevo Spectrum le desbordaba en todos los aspectos: memoria, velocidad de proceso, facilidad de manejo y programación, posibilidades sonoras y gráficas, ampliaciones, etc. En fin, todo lo necesario para tomar contacto con la microinformática a un precio asequible, hasta entonces reservada a los usuarios de equipos profesionales y con economías a prueba de toda solvencia. Las primeras pruebas efectuadas con aquel ordenador despejarían pronto todas nuestras dudas: se trataba de un equipo sencillamente revolucionario.

Por supuesto, no todo eran virtudes y, por ejemplo, el modesto teclado de gomas o la lentitud experimentada en todas las operaciones de transferencia de información entre ordenador y casete, comenzaba a traer de cabeza a todos los usuarios ansiosos de obtener un rendimiento más profesional de su nueva máquina.

Los fabricantes renovaron entonces su oferta y pusieron a nuestra disposición un nuevo medio de almacenamiento de bajo coste y gran velocidad de acceso: el Microdrive.

Poco después aparecería en el mercado el ZX Spectrum +, el cual incorporaba como principal novedad un teclado semiprofesional ampliado que permitía minimizar las pulsaciones necesarias al introducir la información, así como aumentar la precisión y rapidez de estos procesos.

Ya por aquel entonces, primavera de 1985, el mercado apuntaba hacia una nueva generación de microordenadores, los cuales dotaran al usuario de una mayor potencia en el tratamiento de grandes bloques de información.

Los principales fabricantes del momento se lanzaron a una desenfrenada carrera al objeto de cubrir la demanda existente en este terreno, y fruto de ello aparecieron entre otros el Commodore 128 o el Amstrad CPC 6128, con una relación precio/prestaciones, francamente formidable.

Sinclair no podía ser menos y en colaboración con Investrónica efectuó en España durante el SIMO 85 la presentación exclusiva a nivel mundial del nuevo Spectrum 128 K.

LA PRIMERA IMPRESION

Tal acontecimiento pronto suscitó una enconada polémica entre el distribuidor y los primeros usuarios del nuevo prototipo. Opiniones para todos los gustos fueron vertidas en las revistas especializadas en ordenadores Sinclair.

Temas como la absoluta compatibilidad augurada por el distribuidor con los modelos anteriores, el nuevo sistema operativo en modo 128, la ausencia de un medio de almacenamiento rápido y

El nuevo Spectrum 128 K.

fiable que no fuera el sempiterno Microdrive o la mínima información que el usuario recibe en los manuales del equipo no tardaron en ser cuestionados.

Pues bien, tratemos de aclarar todas estas circunstancias comenzando desde el principio. Analicemos cuidadosamente las novedades que incorpora el ZX Spectrum 128 K.

Tras desembalarlo, observamos que la carcasa exterior tiene el mismo aspecto del Spectrum +, de hecho este nombre aparece grabado en la zona superior izquierda, aunque en la parte inferior se añade el distintivo 128 K.

La primera novedad consiste en la aleta de refrigeración exterior situada a la derecha del teclado principal. Su justificación es obvia. El 128 consume algo más que sus predecesores y además, en ellos se observaba una alarmante facilidad para sobrecalentarse tras varias horas de trabajo, y por tanto, la decisión de mejorar la disipación del calor es completamente acertada.

En la parte trasera encontramos los conectores para la fuente de alimentación exterior, la antena de televisión y la tarjeta de expansión, y en el lugar de los enchufes EAR y MIC se ha instalado una salida para monitor RGB.

Estos se hallan situados ahora en el lateral izquierdo de la carcasa y aunque los manuales (más adelante trataremos de ellos) no lo señalan, es posible mantener conectados a la grabadora los dos cables durante las operaciones de lectura/escritura.

Junto a ellos, un conector marcado con RS 232 permite el enlace con impresoras que sigan este protocolo, modems o instrumentos musicales en la norma MIDI.

Finalmente, en el frontal, un pequeño enchufe permite la conexión mediante un cable extensible, del tipo empleado en los teléfonos, del elemento exterior más novedoso: el teclado numérico independiente.

LOS TECLADOS

A primera vista el teclado principal es idéntico al del Spectrum + con sus 58 teclas dispuestas en la misma forma, pero con una diferencia, muchas de ellas están traducidas al español. Tal es el caso de PARAR **(BREAK)**, BORRAR **(DELETE)**, CAMBIO **(SYMBOL SHIFT)**, MODO EXTENDIDO **(EXTEND MODE)**, etc.

Además, la serigrafía ha cambiado en otras muchas de lugar. Por ejemplo, en las teclas 4 y 5 encontramos las letras Ñ y ñ, respectivamente. Para obtenerlas se pulsan simultáneamente éstas y la tecla CAMBIO. Observemos las fotografías y encontraremos bastantes novedades en este aspecto.

Esta circunstancia, a priori, no plantea ningún problema, pues se ha respetado el código ASCII y un programa que antes seleccionaba una op-

Dos octavas contiguas se solapan.

	CANALES					
	TONO			RUIDO		
	A	B	C	A	B	C
CODIGO	1	2	4	8	16	32

								OCTAVA X+1																	
								ESCALA INFERIOR							ESCALA SUPERIOR										
								c	d	e	f	g	a	b	C	D	E	F	b	A	B				
c	d	e	f	g	a	b	C	D	E	F	G	A	B												
ESCALA INFERIOR							ESCALA SUPERIOR																		
OCTAVA X																									

FRECUENCIA	DO	RE	MI	FA	SOL	LA	SI
NOMBRE	C	D	E	F	b	A	B

Códigos para la selección de canales de tono y ruido.

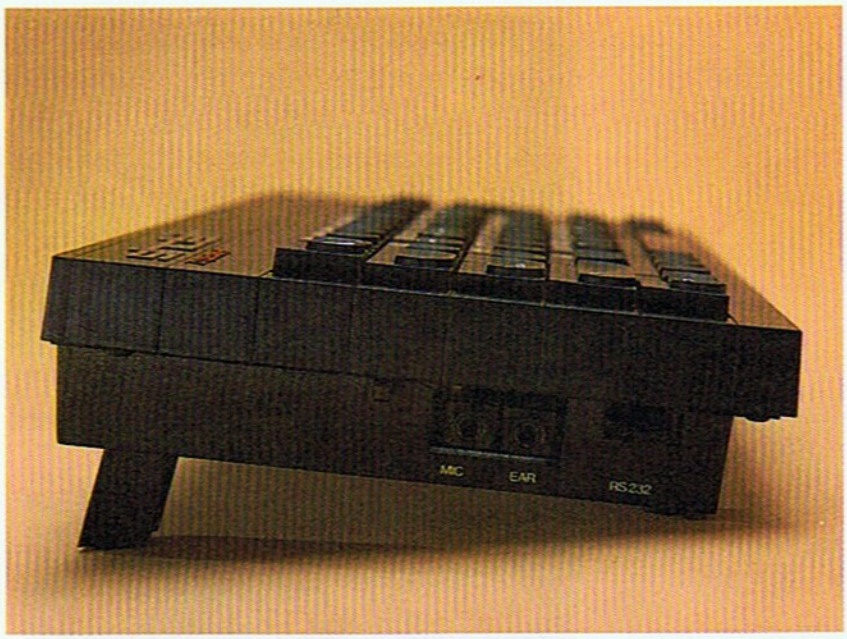
ción tras recibir un carácter dólar ($), seguirá haciéndolo de igual manera.
Los 18 pulsadores del teclado auxiliar constituyen el principal añadido al sistema para simplificar la introducción de datos numéricos y optimizar el trabajo de edición de programas y textos. Nos referiremos a todas sus funciones cuando tratemos el modo de trabajo 128.

EN MARCHA

Cuando conectamos el ordenador a la red aparece sobre la pantalla el nuevo mensaje del fabricante:

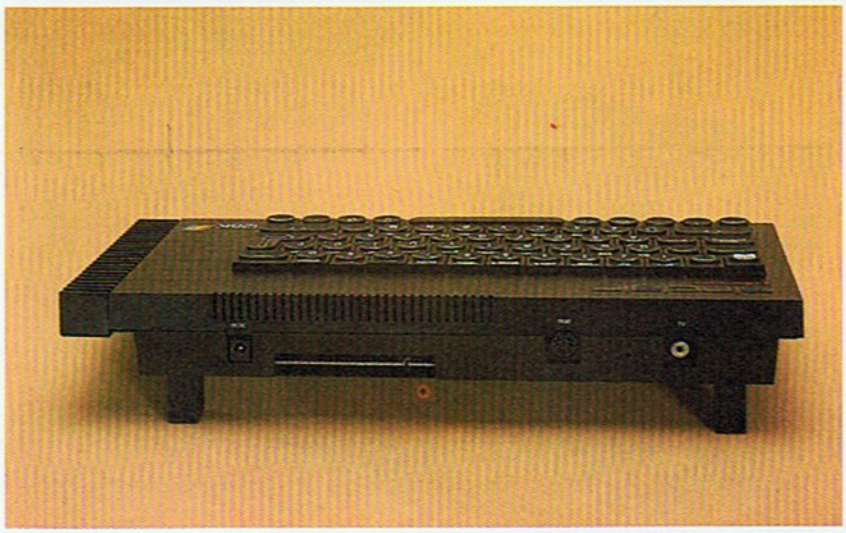
(c) 1985 Sinclair Research Ltd
ESPAÑOL

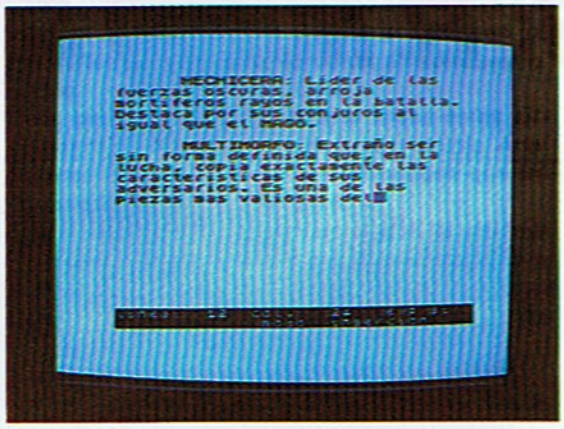
y tras él, un pitido indicando que la inicialización fue correcta. La primera sorpresa es que éste no procede del minialtavoz, sino del televisor.
El buzzer ha sido suprimido y toda emisión de sonido se efectúa vía el altavoz del televisor lo que mejora con creces la altura de la señal audible. Si nuestra pantalla fuera un monitor sin altavoz, podemos recoger el sonido del conector MIC y llevarlo a un amplificador.
A partir de este momento estamos bajo las órdenes del nuevo sistema operativo del 128. En efecto, si pulsamos una tecla aparece en la esquina superior izquierda de la pantalla un cursor cuadrado parpadeante.
Una línea en la parte inferior nos informa constantemente del modo en que nos encontramos: extendido, gráfico, mayúsculas, etc. Al principio estaremos en este último, y letra a letra debemos teclear los comandos del BASIC Sinclair que utilicemos en nuestros programas, y además, en mayúsculas.
De no hacerlo así, el editor considerará las entradas erróneas (no es lo mismo **PRINT** que **print**). Es de especial importancia seleccionar nombres de variables que no contengan entre sus letras palabras del BASIC. Por ejemplo, si tecleamos en una línea **LET CALOR=0**, el error estará garantizado, pues tratará de interpretar el grupo **OR** por separado, y por supuesto, no lo conseguirá.
Una forma de evitarlo consiste en asignar nombres de variables en minúsculas, y la otra, como ya hemos señalado, poner especial atención en no incluir dentro del nombre posibles comandos.
Cada vez que separemos dos sentencias mediante dos puntos (:), el editor automáticamente provoca un salto a la siguiente línea. Por otra parte, si cometemos algún error al efectuar las entradas desde el teclado, en lugar del signo de interrogación (?) parpadeante de todos conocido sobre la posición donde éste ocurrió, ahora encontraremos una especie de cucaracha *(bug,* en su acepción inglesa).
Otra particularidad es que ya no existe el cursor de programa y salvo que nosotros indiquemos lo contrario ejecutando **LIST** o **EDIT** siempre nos encontraremos en la primera página de éste.



En el lateral izquierdo de la carcasa han sido instalados los conectores del casete EAR, MIC y el de impresora RS 232.

Una toma RGB ha sido habilitada para la conexión directa de un monitor en color.





Las dos líneas inferiores del procesador de textos incorporado en el sistema nos informa sobre línea y columna actuales, así como, del modo de escritura implícito.

Pulsando CMND en el teclado independiente o EDITAR en el principal, el cursor se traslada a la última línea de la pantalla. Esta puede emplearse, igualmente, para introducir programas, pero con la salvedad de limitar la entrada a 32 caracteres.

Aquí nos está permitido hacer uso de los nuevos comandos del editor:

— EDIT o E seguido por un número de línea muestra el programa a partir de ésta con el cursor preparado al principio de ella para comenzar la edición.

— NUM o N y a continuación dos números separados por una coma, indican al editor la nueva base y el incremento entre líneas elegido para efectuar la renumeración de todo el programa BASIC almacenado en ese momento en la memoria. En este sentido se echa de menos un comando, el cual permitiera la renumeración parcial, por bloques y no siempre total.

— BORRAR o B con dos parámetros numéricos a continuación eliminará del programa las líneas comprendidas entre ambos, incluidas éstas también.

En el teclado numérico auxiliar están previstas algunas funciones que permiten corregir los programas con cierta comodidad. A parte de cuatro teclas de cursor, las cuales facilitan el movimiento en todos los sentidos por el listado, es factible avanzar con una sola pulsación al principio o final de una misma línea o de la siguiente.

Dos teclas de borrado para eliminar el carácter situado bajo el cursor o el ubicado a su izquierda, más un duplicado de ENTER completan las teclas de edición de programas.

DISCO RAM

De los 128 Kbytes disponibles en este modelo, la cantidad aprovechable para la programación en BASIC sigue manteniéndose inalterada respecto a sus predecesores, es decir, casi 41 K. El resto de memoria adicional sólo es accesible desde código máquina.

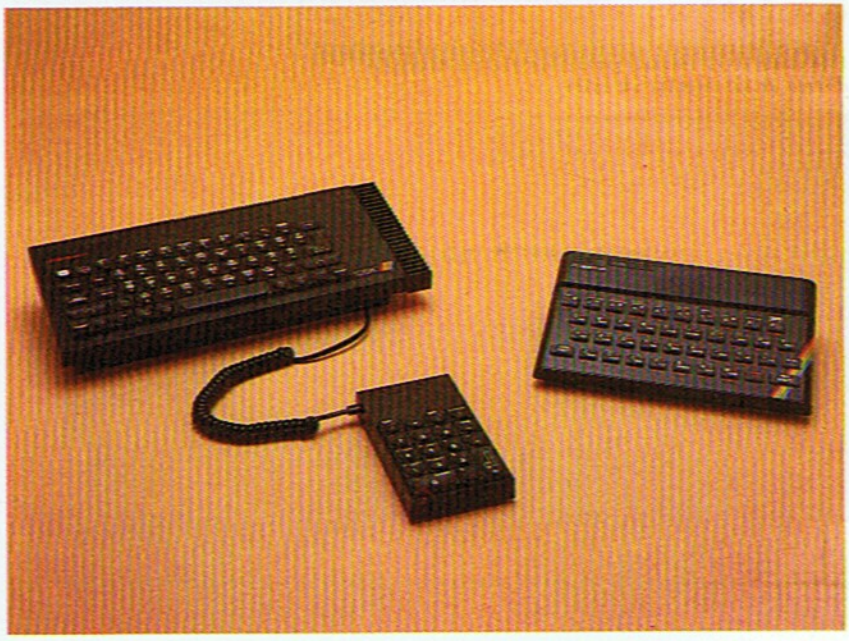
Pero además, ofrecen una posibilidad adicional: la de servir de almacenamiento de datos y programas. Para hacer uso de ella deberemos encontrarnos en modo 128. Los comandos son los mismos que los de casete con el añadido del símbolo de admiración (!).

Una ventaja fundamental frente a otros soportes de información es la transferencia prácticamente instantánea de ésta y la total fiabilidad (no es necesario verificar la información una vez almacenada). Pero por contra perdemos su contenido cuando desconectamos el ordenador o pasamos al modo 48 K.

No obstante en programas que manejen un volumen apreciable de datos, el cual nos obligue a continuas operaciones de carga o grabación de éstos, el Disco RAM será una inestimable ayuda, siendo posible visualizar en cualquier momento su contenido mediante el comando **CAT!**.

El paso al modo 48 K se efectúa ejecutando como comando directo o dentro de un programa la orden **SPECTRUM**. Tras esto, el teclado auxiliar

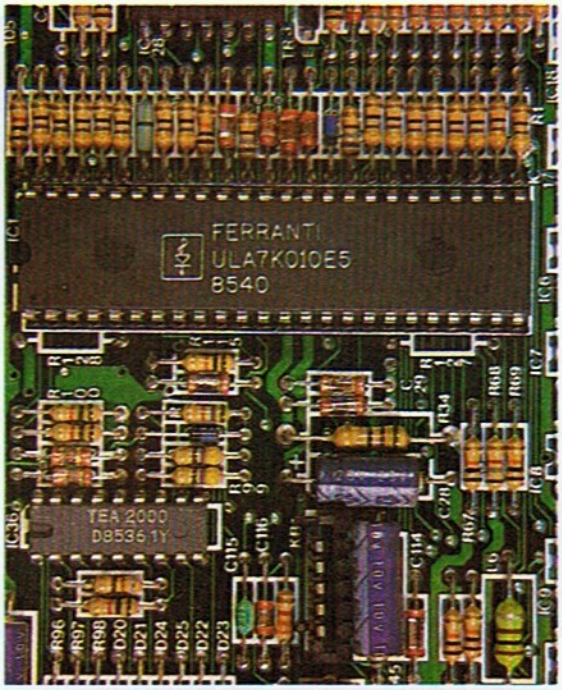
El 128 K pretende venir a sustituir a su antecesor: el sorprendente Spectrum 48 K.



queda deshabilitado y podemos desconectarlo si solamente vamos a trabajar en este sistema. A partir de entonces el ordenador se comporta como si de un Spectrum + se tratara.
El contenido de la memoria no se borra por esta operación, si bien todos los comandos especiales comentados anteriormente no serán operativos, y en algunos casos conducirán al mensaje de error "NO EXISTE EN BASIC". Tanto éste, como los restantes mensajes de la ROM han sido traducidos al castellano, y así, en vez del habitual *Scroll?* aparecerá ¿SIGO?, por ejemplo.
El paso 128-48 se realiza también tecleando letra a letra **RUN USR 0**, pero en este caso perderemos el contenido de la memoria. El salto inverso, es decir, de modo 48 a modo 128 no está permitido y nos veremos obligados a desconectar la alimentación o pulsar el botón de RESET para conseguirlo.

PROCESANDO TEXTOS

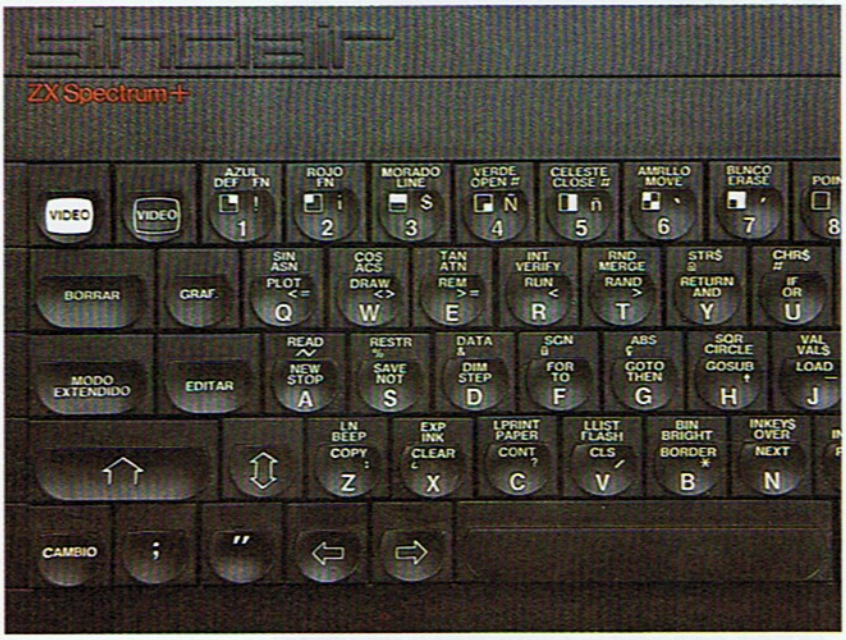
Entre las novedades incorporadas en el 128 se encuentra presente el acceso mediante el comando **EDIT**, y tras él una variable alfanumérica, al procesador de documentos residente en el sistema.
Los distribuidores anuncian entre sus virtudes la posibilidad de creación de largos documentos, la facilidad de corrección de errores y desplazamiento por todo el texto, así como la impresión de éste a base de cómodos comandos y manejando el teclado auxiliar.
Pues bien, en la prueba efectuada encontramos que cada línea contiene como máximo 32 caracteres. No es posible justificar los textos por la derecha. La búsqueda de palabras y el movimiento de bloques no se ha previsto.
Las vocales acentuadas no están en el juego de caracteres de serie. No obstante, es posible crearlas como gráficos definidos y teclearlas cuando sea necesario en este modo.
Si en un texto de 320 líneas (la mitad de la capacidad, aproximadamente, disponible en los procesadores para Spectrum comerciales), le ordenamos mediante la tecla correspondiente del teclado auxiliar que salte de la primera línea a la última, con cierta sorpresa comprobamos que esta operación se realiza en aproximadamente 56 segundos.
Cuando se trata de grabar el documento, la única opción posible es a cinta, pues el paso a Microdrive provoca el error NO EXISTE EN BASIC.



La nueva ULA de 48 pines: 8 más que en las versiones anteriores.

En resumen, en la creación de documentos cortos y con pocas pretensiones en cuanto a formato y longitud, su utilidad es dudosa, pero en el caso de grandes documentos la velocidad con que los trata lo convierte en inapropiado para esta labor.

El teclado auxiliar facilita la edición de programas y el control del procesador de textos.



Las inscripciones de muchas teclas ha sido traducida al castellano y entre los caracteres de este idioma aparecen las eñes.

Sorprende, por tanto, que en la publicidad se oferte como una alternativa en el campo del proceso de textos, cuando éste se haya implementado con tan baja calidad.

EL SONIDO EN EL 128

Frente al proceso de textos, donde nuestras quejas quedan más que justificadas, el 128 dispone de un nuevo PSG *(Programmable Sound Generator,* generador de sonidos programable), el cual amplía con creces las posibilidades musicales ya conocidas de los modelos anteriores.

Desde BASIC es controlable mediante **BEEP** (en esta misma sección está recogida toda la información referente a él), y el nuevo comando **PLAY**, solamente operativo cuando nos encontramos en modo 128.

PLAY puede controlar simultáneamente tres canales de sonido independientes denominados A, B y C. La asignación de la música o efecto sonoro a cada canal se realiza mediante cadenas alfanuméricas, es decir, el formato general de este comando tiene el siguiente aspecto: **PLAY cadena1$, cadena2$, cadena3$.**

En la construcción de cadenas actúan varios nuevos comandos, los cuales han sido resumidos en la tabla adjunta. Comentemos los más importantes:

— M permite la selección de canal para la emisión de tono, ruido o mezcla de ambos. Por defecto, los canales A, B y C están preparados como generadores de tono, pero según el valor situado tras el comando M, podemos modificar esta circunstancia efectuando combinaciones de los valores señalados en la tabla.

— Cada nota se obtiene asignándole el nombre reservado para ella según la escala de la figura.

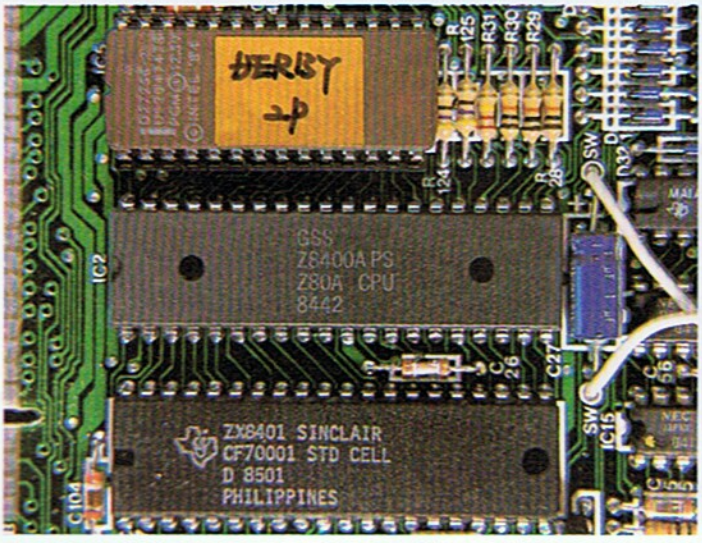
— O selecciona la octava mediante el parámetro situado a continuación. Este puede variar de 0 a 8, siendo la más grave la marcada con O0 y la más aguda la O8.

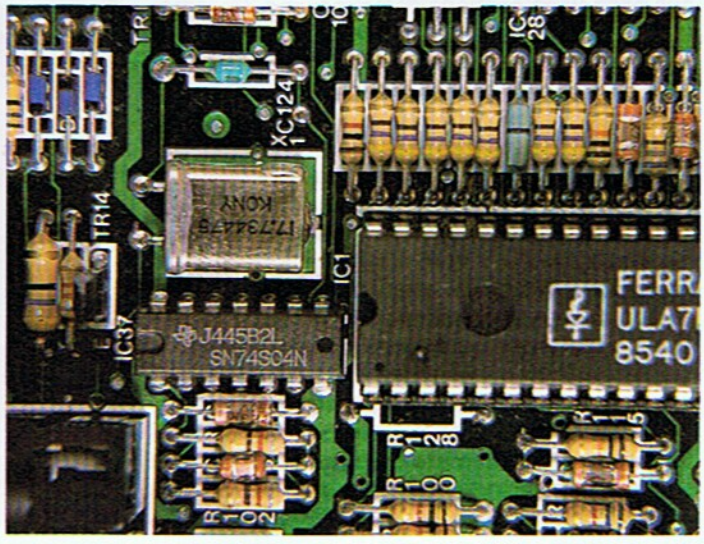
— Un número entre 1 y 12 colocado antes de una nota establece su duración.

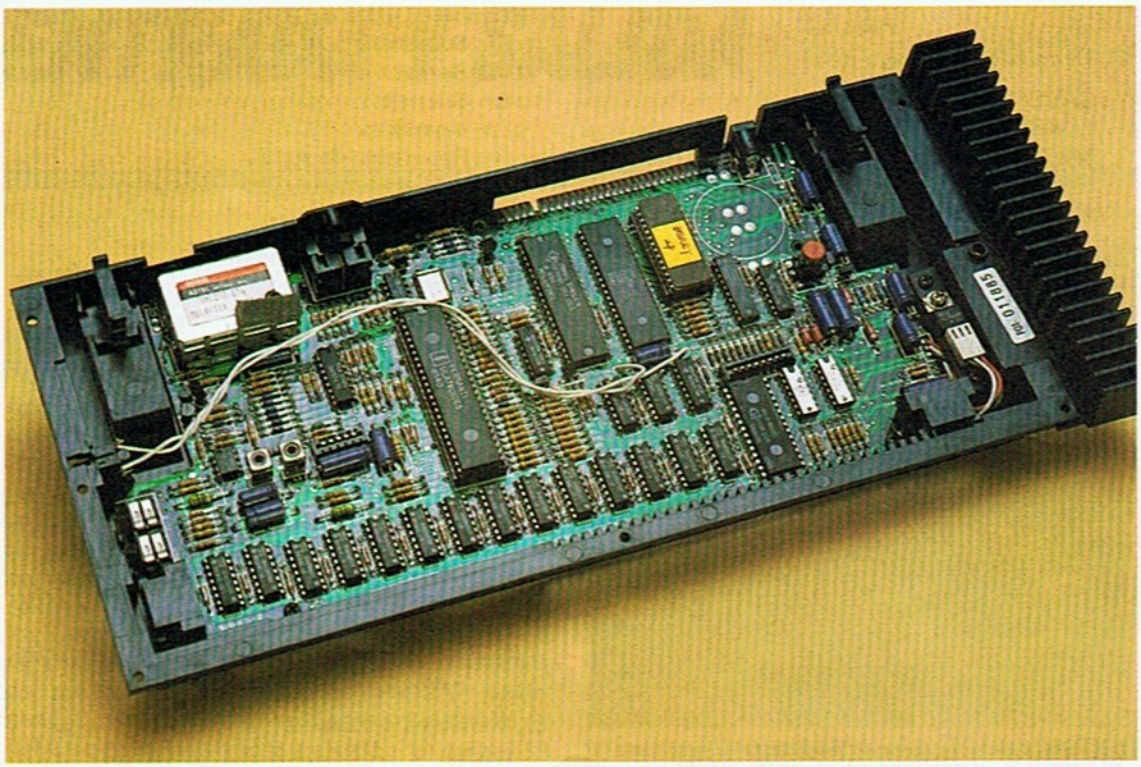
— N actúa de separador de comandos cuando la unión de éstos pueda dar lugar a confusión.

Observemos la tabla de comandos y comprobaremos que además de los anteriores parámetros

El cristal de cuarzo ha sido sustituido por una nueva versión regulada a 17.734475 MHz.







Levantando la carcasa aparece ante nuestros ojos el nuevo circuito impreso y sus componentes.

destinados al control del volumen, velocidad de ejecución, repetición de grupos de notas, etc., confieren al Spectrum 128 unas posibilidades sonoras formidables.

CONCLUSIONES

Entre las virtudes del nuevo sistema quedan contrastadas sus posibilidades musicales o la mejora en el proceso de programas con gran volumen de datos de entrada/salida valiéndonos del Disco RAM.

El nuevo teclado auxiliar simplifica la edición de programas y la introducción de datos numéricos, así como el empleo del ordenador como calculadora. En este sentido, el sistema, tras una operación, retiene en su memoria el resultado por si el usuario necesita encadenarlo con el siguiente.

Se echa de menos un BASIC ampliado que mejore la capacidad de proceso y la potencia operativa general del nuevo ordenador, así como, un medio de almacenamiento masivo de información de alta velocidad de acceso y fiabilidad.

La traducción de los mensajes de la ROM al castellano facilita al operador profano en inglés la comprensión de éstos, si bien, parece un síntoma de improvisación encontrar como traducción de "Out of screen", "FUERRA DE LA PANTALLA", donde efectivamente, se ha deslizado una "R" de más.

El procesador de textos incorporado constituye el punto más débil de todo el sistema, y por supuesto, quien desee emplear su nuevo ordenador en este campo, más vale que vaya pensando en alguno de los comerciales.

El tema de la total compatibilidad a nivel de software y hardware conviene ponerlo cuando menos en tela de juicio. En principio todo el software disponible para el Spectrum + es implementable en el 128 procesándolo en modo 48 K, aunque sólo el tiempo confirmará o rebatirá tal posibilidad.

A nivel de hardware, la cuestión es muy diferente, y todos aquellos periféricos que para su funcionamiento aprovechan la primitiva rutina de inicialización de la ROM son totalmente incompatibles. Tal es el caso del SOUND JOYSTICK INTERFACE de INVESTRONICA. El poseedor de uno de estos dispositivos, puede ir pensando en adquirir un nuevo modelo.

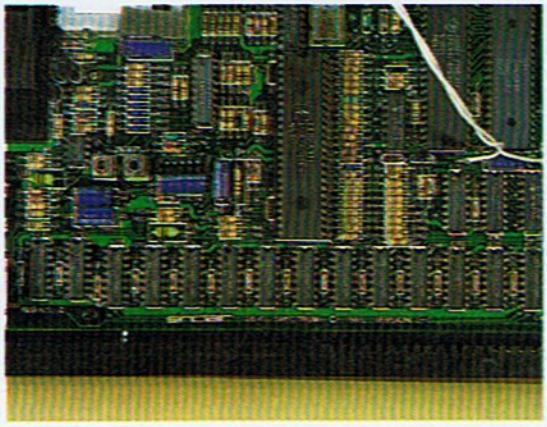
En resumen, un ordenador en el cual ha primado en su diseño la ampliación de las posibilidades lúdicas del Spectrum por aprovechamiento de una mayor capacidad de memoria, pero inadecuado para todo usuario que con el cambio de modelo pretenda trabajar bajo una óptica algo más profesional.

TABLAS

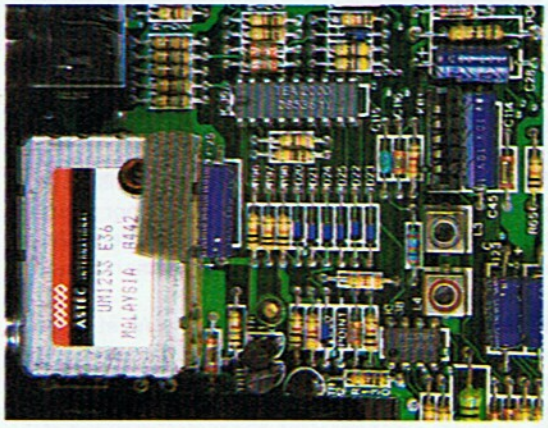
1) Comandos del Disco RAM

SAVE!"nombre"
LOAD!"nombre"
MERGE!"nombre"
CAT!
ERASE!"nombre"

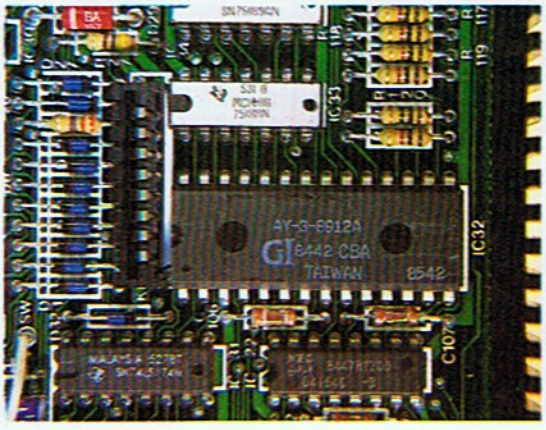
Los pequeños chips bajo la ULA soportan los 112 Kbytes de memoria RAM.



Los nuevos circuitos cerca del modulador se encargan de enviar las señales sonoras hacia el altavoz del televisor.



El chip AY-3-8912A es el nuevo PSG (Program Sound Generator) encargado de la producción de sonidos a través de tres canales independientes.



2) Símbolos en la creación de sonidos

) Sostenido
$ Bemol
& Silencio
_ Ligaduras
() Repetición de notas
!! Comentarios

3) Comandos de sonido

M- Selección de canal.
O- Selección de octava.
1-12- Duración de las notas.
N- Separador de comandos.
V- Volumen de las notas (0-15).
W- Selección de envolvente.
X- Longitud de la envolvente.
U- Selección de canales afectados por el generador de envolventes.
T- Tempo (bpm o golpes por minuto).
H- Corta las repeticiones indefinidas.
a-g Escala inferior de notas.
A-G Escala superior de notas.
Y- Habilita la salida MIDI (1 a 16).
Z- Cambia la programación del sintetizador.

5) Mensajes del casete

PREPARE LA CINTA Y PULSE ENTER
PROGRAMA:
MATRIZ NUM.:
MATRIZ CARACT.:
BYTES:

4) Mensajes del 128

0 OK
1 NEXT SIN FOR
2 VARIABLE INDEFINIDA
3 INDICE ERRONEO
4 SIN MEMORIA
5 FUERRA DE PANTALLA
6 NUMERO MUY ALTO
7 RETURN SIN GOSUB
8 FIN DE FICHERO
9 SENTENCIA STOP
A ARGUMENTO ERRONEO
B ENTERO FUERA DE RANGO
C NO EXISTE EN BASIC
D BREAK/CONT SIGUE
E FALTAN DATOS
F NOMBRE/FICHERO ERRONEO
G NO HAY SITIO
H STOP EN INPUT
I FOR SIN NEXT
J DISPOSITIVO E/S ERRONEO
K COLOR INVALIDO
L BREAK EN PROGRAMA
M RAMTOP MALA
N SENTENCIA PERDIDA
O CORRIENTE INVALIDA
P FN SIN DEF
Q PARAMETRO ERRONEO
R ERROR DE CARGA
¿SIGO?

ENTRADA Y SALIDA

El sistema de comunicaciones que un microprocesador debe mantener con los periféricos que le rodean, es uno de los más importantes factores considerado por los fabricantes de un ordenador, cuando han de tomar la decisión de incorporarlo a sus equipos.

En el caso del Z 80, la interacción con el teclado, la ULA, la cual controla la imagen que parte hacia el televisor, el casete, las impresoras, etc. se encuentra satisfactoriamente gestionada a través de las rutinas almacenadas en la ROM de nuestro Spectrum.

Pero no olvidemos que cuando un microprocesador necesita enviar o recoger información destinada o procedente de un dispositivo exterior, lo hace siempre byte a byte a través de los canales habilitados a tal efecto del bus de datos.

Este como sabemos, está formado por 8 líneas, y a través de cada una de ellas puede circular un bit cada vez, es decir, hablamos de un bus de datos de 8 bits o un byte, el cual podríamos decir que comienza en el microprocesador y termina en el puerto (port) habilitado para el periférico.

IN Y OUT

IN y **OUT** son las dos instrucciones preparadas por los fabricantes del **Z 80** encargadas de gestionar cualquier transferencia de datos entre microprocesador y periférico. La CPU no necesita saber como funciona un dispositivo exterior a ella en detalle, sino tan solo, de donde **(IN)** o a donde **(OUT)** recoger o enviar la información que con él intercambia.

MNEMONICO	CODIGO MAQUINA	REGISTRO F 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Nº BYTES	CICLOS MAQ.	CICLOS RELOJ	NOTAS
DI	1 1 1 1 0 0 1 1	•	•	•	•	•	•	•	•	1	1	4	
EI	1 1 1 1 1 0 1 1	•	•	•	•	•	•	•	•	1	1	4	
IM0	1 1 1 0 1 1 0 1 0 1 0 0 0 1 1 0	•	•	•	•	•	•	•	•	2	2	8	
IM1	1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0	•	•	•	•	•	•	•	•	2	2	8	
IM2	1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0	•	•	•	•	•	•	•	•	2	2	8	
RETI	1 1 1 0 1 1 0 1 0 1 0 0 1 1 0 1	•	•	•	•	•	•	•	•	2	4	14	
RETN	1 1 1 0 1 1 0 1 0 1 0 0 0 1 0 1	•	•	•	•	•	•	•	•	2	4	14	

Instrucciones de control de interrupciones.

Este lugar es lo que antes hemos denominado puerto, o port en su acepción inglesa, y puesto que el Z 80 no le interesa como llega o sale de allí cada byte desde/hacia el periférico, envía o recoge un octeto, de uno en uno, cada vez que sea necesario. Y éste es, precisamente, el trabajo encomendado a las instrucciones **IN** y **OUT**.
El formato más sencillo de las instrucciones de entrada **(IN)** de datos hacia el microprocesador es:

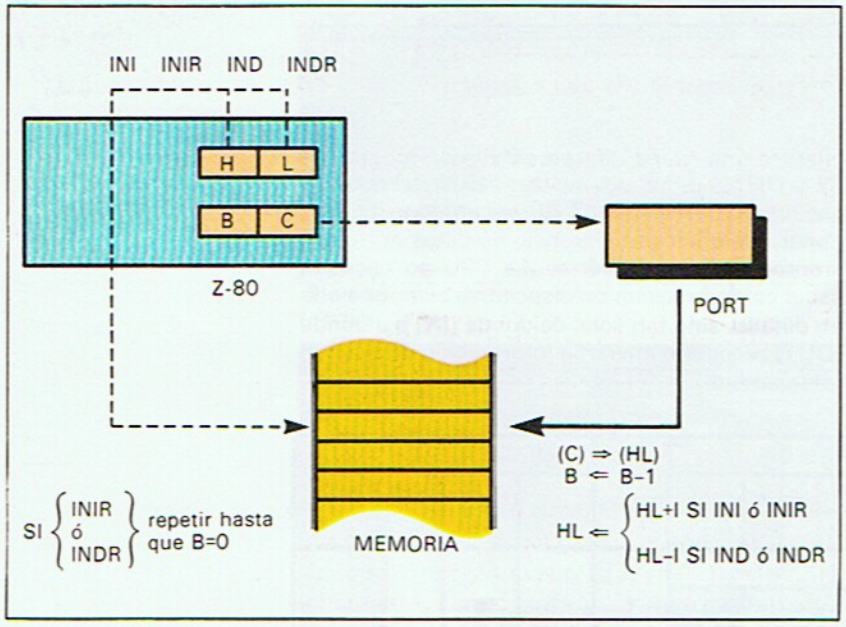
IN DESTINO, PORT

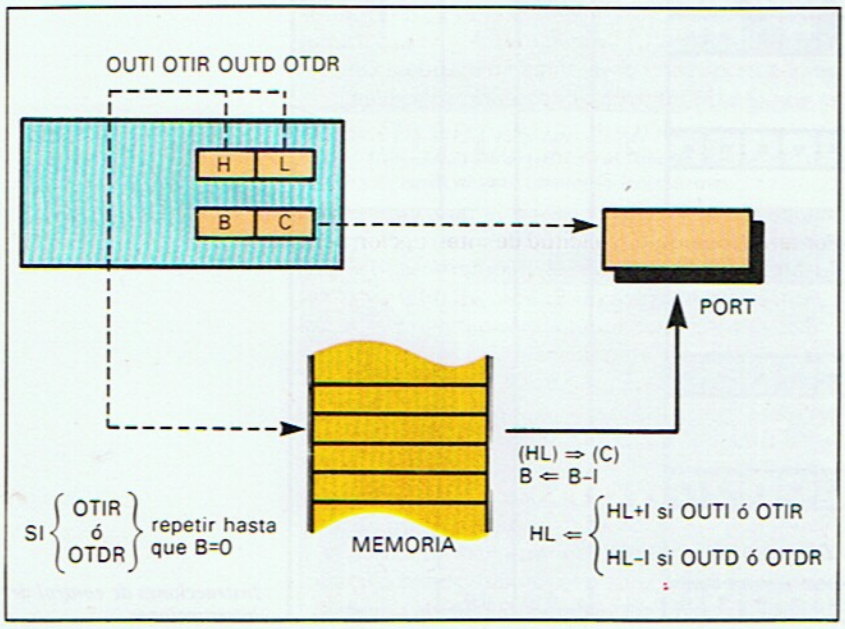
donde DESTINO puede ser cualquiera de los registros de 8 bits A, B, C, D, E, H o L y PORT, el puerto del cual nosotros necesitemos recavar la información definido por el contenido del par BC en las instrucciones IN r,(C) y por el par An en IN A,(n).
Respecto de las instrucciones de salida **(OUT)**, podemos efectuar idénticas apreciaciones, pero considerando ahora que la operación es inversa, es decir, enviamos a un puerto determinado el byte almacenado en el registro ORIGEN del microprocesador. El formato será, por tanto:

OUT (PORT), ORIGEN

donde tanto PORT como ORIGEN, quedan definidos exactamente igual que en las instrucciones de entrada pero, por supuesto, el sentido de la transferencia de información es opuesto (del Z 80 al periférico).
De forma similar a lo que ocurría con las instrucciones de búsqueda y transferencia de bloques, las de entrada/salida pueden funcionar semiautomáticamente o de manera totalmente automática para recoger o enviar la información contenida o destinada a un determinado bloque de memoria desde/a un port específico.
En estos casos el par HL debe señalar al primer octeto del bloque (instrucciones INI, INIR, OUTI y OTIR) o al último (caso de IND, INDR, OUTD y OTDR) de donde o hacia donde recoger/enviar la información, puesto que en las primeras se incrementa de uno en uno y en las segundas se decrementa en la misma cantidad cada vez que se realiza una lectura/escritura del port correspondiente.
El registro B es utilizado como contador, y en todas estas instrucciones se decrementa en uno por cada lectura o escritura que se realice del puerto especificado por el registro C.
Si las instrucciones son de mecanismo automático, la transferencia de datos entre ordenador y periférico se ejecuta una y otra vez hasta que el contenido del registro B sea 0. En las figuras hemos representado esquemáticamente todos estos procesos.

Mecanismo de las instrucciones de entrada y salida.





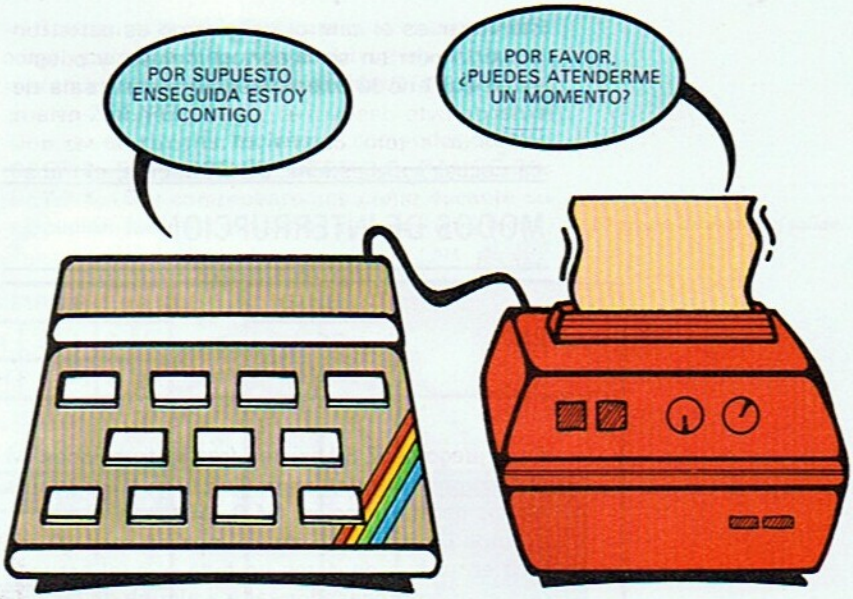
INTERRUPCIONES

Una interrupción es una señal enviada por un periférico hacia el microprocesador con la intención de que éste abandone cualquier trabajo que estuviera realizando y atienda su solicitud.
Cuando esta circunstancia se produce, la CPU completa la ejecución de la instrucción en curso en ese instante y salta a una dirección de memoria donde encontrará almacenada la subrutina de

tratamiento de la interrupción, anotando previamente en el stack, donde paralizó el programa principal.
Una vez concluida la rutina de interrupción retorna al punto de salto, siempre que al final de esta hayamos incluido una instrucción de retorno, y continúa implementando el resto del programa.
Sobre este particular conviene puntualizar que, en principio, bastaría colocar al final de la subrutina una instrucción de retorno de las ya conocidas, por ejemplo, RET. Pero solamente las de retorno de interrupción (RETI y RETN), garantizan que el microprocesador encontrará restablecidas las condiciones de partida previas, al tratamiento de la solicitud del periférico en cuestión.
En el Z 80, microprocesador del Spectrum, dos de sus patillas están dispuestas para recibir las peticiones de interrupción por parte de un dispositivo exterior.
El pin 17 denominado NMI *(Non Maskable Interruption)* se encargaría de recibir las interrupciones no enmascarables y hemos puntualizado «se encargaría», pues el Spectrum no utiliza esta línea.
El hecho es que las instrucciones NMI, de haberlas, gozarían de prioridad absoluta y sería imposible desactivarlas por parte del programador. El microprocesador saltaría siempre a la rutina ubicada a partir de la dirección 0066h (102 decimal) y ésta le indica que provoque la reinicialización del sistema sin llegar nunca a la instrucción RETN (retorno de interrupción no enmascarable) con la que finaliza dicha subrutina.
Visto así, parece que este tipo de interrupciones carece de utilidad práctica (prueba a ejecutar **RANDOMIZE USR 102**). Desde el punto de vista de la programación de nuestro Spectrum, probablemente sí.
Pero algunos periféricos de los existentes en el mercado son capaces de bloquear la ROM de forma que a partir de la dirección 0066h, el Z 80 encuentre una rutina diferente de la preprogramada en el sistema, adquiriendo de esta manera prioridad frente a cualquier otro dispositivo.

Códigos de las instrucciones de interrupción.

MNEMONICO	HEXA-DECIMAL	DECIMAL
DI	F3	243
EI	FB	251
IM0	ED 46	237, 70
IM1	ED 56	237, 86
IM2	ED, 5E	237, 94



Protocolos.

INTERRUPCIONES ENMASCARABLES

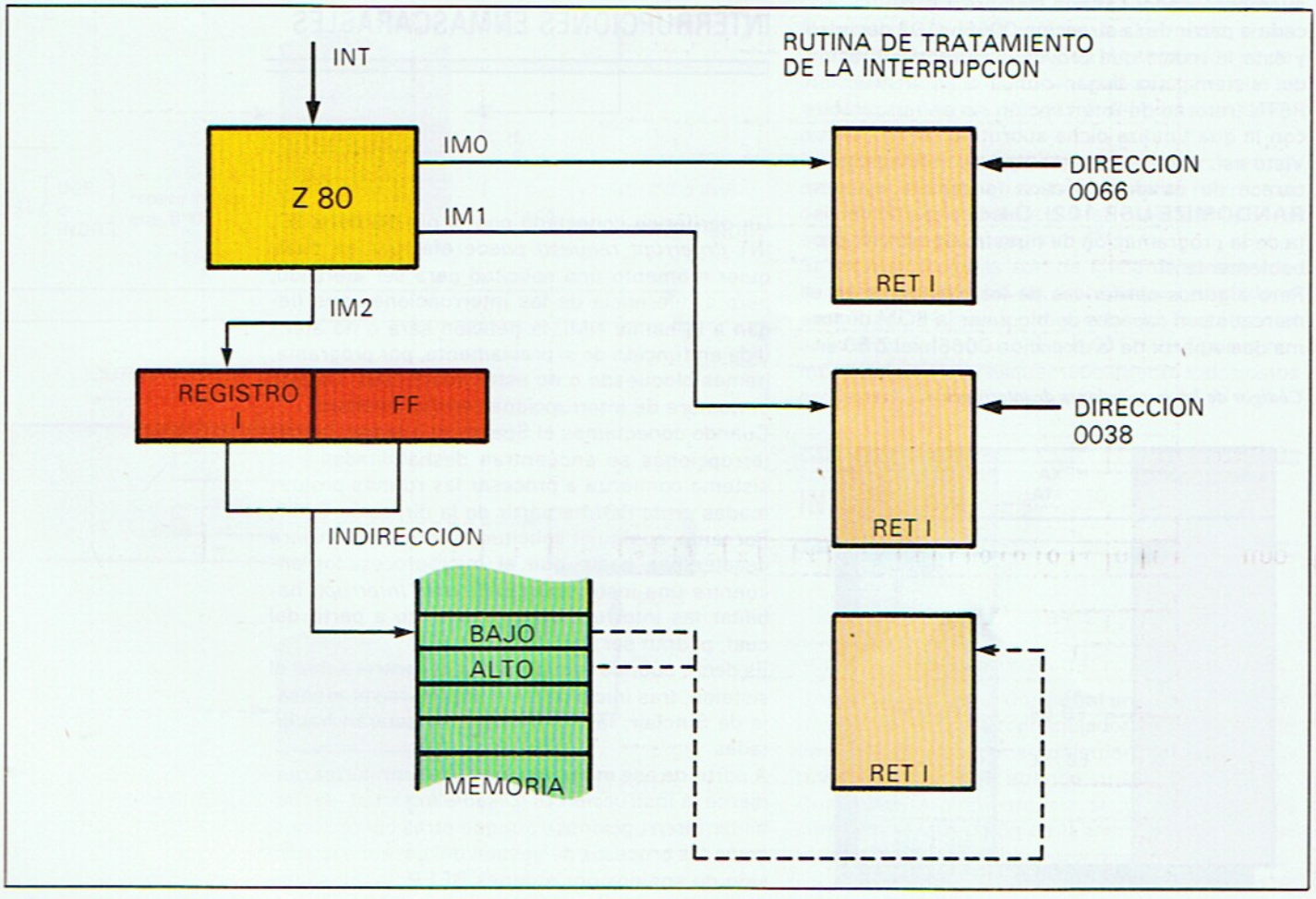
Un periférico conectado con el pin 16 del Z 80, INT *(Interrupt request)* puede efectuar en cualquier momento una solicitud para ser atendido, pero a diferencia de las interrupciones que llegan a la patilla NMI, la petición será o no atendida en función de si previamente, por programa, hemos bloqueado o no estas solicitudes. De ahí, el nombre de interrupciones enmascarables.
Cuando conectamos el Spectrum a la red, las interrupciones se encuentran deshabilitadas y el sistema comienza a procesar las rutinas programadas en la ROM a partir de la dirección 0000. Por tanto, cualquier solicitud de interrupción será desatendida hasta que el microprocesador encuentre una instrucción EI *(Enable Interrupt,* habilitar las interrupciones), momento a partir del cual, podrán ser atendidas.
Es decir, cuando recuperamos el control sobre el sistema, tras inicializarse e imprimirse el mensaje de Sinclair, las interrupciones estarán habilitadas.
A partir de ese momento, podemos inhibirlas mediante la instrucción DI *(Disable Interrupt,* deshabilitar interrupciones), aunque otras operaciones, como los procesos de gestión del casete o la emisión de sonidos por órdenes **BEEP**, las deshabi-

litarán, pues el control del tiempo es parte fundamental de su funcionamiento y parece lógico evitar que el Z 80 sea molestado mientras los ejecuta.

MODOS DE INTERRUPCION

En el juego del Z 80 existen tres instrucciones IM *(Interruption Mode)* las cuales fijan otros tantos modos de interrupción: IM 0, IM 1 e IM 2.
Cuando un periférico interrumpe en modo 0 coloca en el bus de datos un byte el cual provoca que el microprocesador salte a alguna de las primeras direcciones de la página cero. En el Spectrum, este modo no se utiliza, salvo en el caso de un RESET, tras el cual, la CPU comienza a procesar la rutina de inicialización.
El modo 1 es similar al anterior, pero en este caso la dirección de salto es constante (0038h). Es el modo habitual en el Spectrum, pues a partir de dicha dirección se encuentra la rutina de tratamiento del teclado, y como ya sabemos, la ULA cursa 50 veces cada segundo una solicitud de interrupción para que este sea examinado.
Finalmente, mediante la instrucción IM 2, se accede al modo de interrupción 2 el cual, desde el punto de vista de la programación es el más interesante para nosotros. Cuando una solicitud es efectuada al microprocesador por este sistema, el Z 80 conforma una dirección absoluta de salto entre el contenido del registro I (parte alta) y el byte entregado por el periférico (parte baja) al bus de datos.
La dirección así formada (byte colocado en el bus de datos + 256 × contenido de I) señala a una posición de memoria determinada. Ahora la CPU lee el contenido de ésta y de la siguiente celda, y con estos dos octetos forma la dirección absoluta donde encontrar la subrutina de tratamiento de la interrupción (1.ª posición + 256 × siguiente), método conocido técnicamente como de «indirección».
Cuando en un programa nosotros hacemos uso de este mecanismo de interrupción, al no existir el periférico que lo solicite, el Spectrum deposita en el bus de datos, siempre el valor FFh. Como

Los modos de interrupción.



vemos, el valor que nosotros asignemos al registro I es determinativo.
En principio, su valor podrá ser cualquiera con tal que la posición de memoria obtenida contenga los valores que finalmente conformarán la dirección efectiva de tratamiento de la interrupción.
Como demostración hemos preparado una pequeña rutina la cual pasa del modo IM1, habitual en el Spectrum, al IM2. Cada vez que se genera una interrupción para ser analizado el teclado, se producen ahora unos efectos de borde a través del puerto 254 (FEh).
Una vez en marcha, tecleemos comandos del tipo **BEEP 2,-5: PAUSE 20: BEEP 3,0: PAUSE 30: BEEP 4,10** y comprobaremos como durante su ejecución las interrupciones son deshabilitadas.

El grupo de entrada/salida.

MNEMONICO	CODIGO MAQUINA	REGISTRO F 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Nº BYTES	CICLOS MAQ.	CICLOS RELOJ	NOTAS
INA,(n)	1 1 0 1 1 0 1 1 x n x	•	•	x	•	x	•	•	•	2	3	11	
INr,(c)	1 1 1 0 1 1 0 1 0 1 r r r 0 0 0	↕	↕	x	↕	x	p	0	•	2	3	12	
INI	1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 0	x	① ↕	x	x	x	x	1	•	2	4	16	
INIR	1 1 1 0 1 1 0 1 1 0 1 1 0 0 1 0	x	1	x	x	x	x	1	•	2	5/4	21/16	
IND	1 1 1 0 1 1 0 1 1 0 1 0 1 0 1 0	x	① ↕	x	x	x	x	1	•	2	4	16	
INDR	1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0	x	1	x	x	x	x	1	•	2	5/4	21/16	
OUT(n),A	1 1 0 1 0 0 1 1 x n x	•	•	x	•	x	•	•	•	2	3	11	
OUT(c),r	1 1 1 0 1 1 0 1 0 1 r r r 0 0 1	•	•	x	•	x	•	•	•	2	3	12	
OUTI	1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1	x	① ↕	x	x	x	x	1	•	2	4	16	
OTIR	1 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1	x	1	x	x	x	x	1	•	2	5/4	21/16	
OUTD	1 1 1 0 1 1 0 1 1 0 1 0 1 0 1 1	x	① ↕	x	x	x	x	1	•	2	4	16	
OTDR	1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1	x	1	x	x	x	x	1	•	2	5/4	21/16	

REG	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

1 Significa que el indicador Z será colocado a 1, si 3-1 es cero; en caso contrario será puesto a 0.
Cuando en CICLOS se dan dos cantidades la primera indica ciclos si B≠0 y la segunda ciclos cuando B=0.

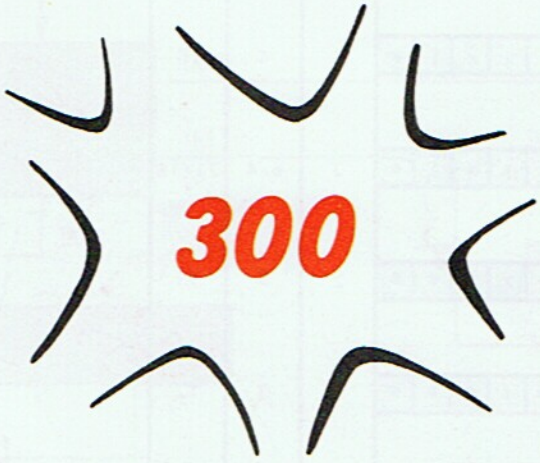
EMBOSCADA

Emboscada es un juego de habilidad con el cual pasaremos unos momentos agradables, sin duda una buena despedida.

El objetivo del juego es dirigir una calabaza mágica, porque si no añadimos lo de mágica no sabemos como justificar que una calabaza ande sola, por el recinto en el cual está prisionera, sin que tropiece ni resulte atrapada por los cuadros centelleantes que irán surgiendo a su paso. Para hacer más sencilla esta misión, cada trescientos puntos desaparecerán todos los obstáculos, recomenzando el ciclo.

Para el desplazamiento de nuestra calabaza se utilizarán las siguientes teclas:

"O" ..... "Izquierda"
"P" ..... "Derecha"
"Q" ..... "Arriba"
"A" ..... "Abajo"

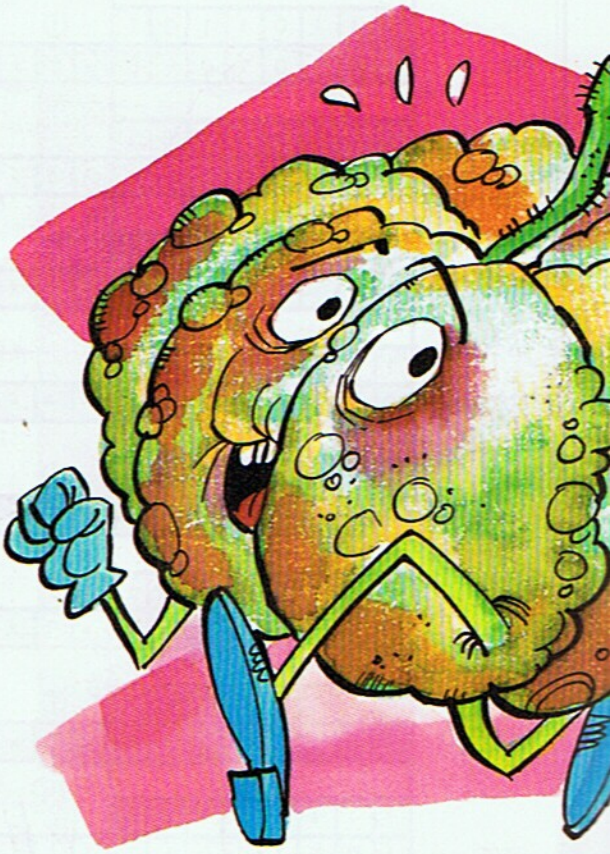


Para hacer más sencillo el juego, cada 300 puntos desaparecerán todos los obstáculos, recomenzando el ciclo.

Nuestra calabaza posee tres vidas, gracias a lo cual es posible conseguir una buena puntuación. Para aquellos que sean extremadamente malos en este tipo de juegos, cada vez que aparezca una calabaza, o por cada punto se le anotarán diez de propina. De esta forma es imposible finalizar la partida con cero puntos.

INTRODUCCION DEL PROGRAMA

A la hora de introducir el listado del programa, debemos destacar que los caracteres que figuran en el listado con subrayado simple, corresponden a los caracteres gráficos de las teclas afectadas,



¡!

A la hora de introducir el listado del programa, debemos destacar que los caracteres que figuran en el listado con subrayado simple, corresponden a los caracteres gráficos de las teclas afectadas, y los de doble subrayado, a las mayúsculas de los caracteres gráficos correspondientes.

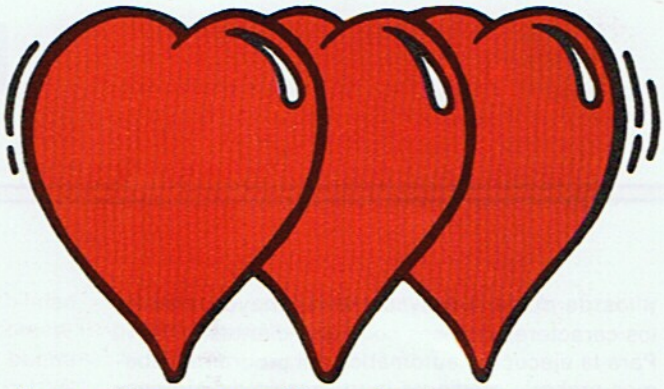
*

Para la ejecución automática del programa debemos grabarlo mediante la utilización del siguiente comando: **SAVE "EMBOSCADA" LINE 1**. Si prefiriésemos la grabación sin autoejecución tecleríamos la siguiente orden **SAVE "EMBOSCADA"**.

y los de doble subrayado, a las mayúsculas de los caracteres gráficos correspondientes.
Para la ejecución automática del programa debemos grabarlo mediante la utilización del siguiente comando: **SAVE "EMBOSCADA" LINE 1.** Si prefiriésemos la grabación sin autoejecución tecleríamos la siguiente orden **SAVE "EMBOSCADA".**
En cuanto a los literales **INV.** y **TRUE** que aparecen subrayados y entre corchetes, corresponden a los caracteres de control de video inverso o **INVERSE VIDEO (CAPS SHIFT + 4)** y video normal o **TRUE VIDEO (CAPS SHIFT + 3).**

El objetivo del juego es dirigir la calabaza por el recinto en el cual está prisionera, sin que tropiece ni resulte atrapada.

```
  1 REM ************************
  2 REM * C.E.IGLESIAS MTNEZ. *
  3 REM ************************
  4 REM * ATRAPADA   (C) 1985 *
  5 REM ************************
  6 REM *     ADAPTACION:      *
  7 REM ************************
  8 REM * J.M.MAYORAL SERRANO *
  9 REM ************************
 10 REM * EMBOSCADA  (C) 1986 *
 11 REM ************************
 15 POKE 23658,8
 20 LET CONTA=0
 30 GO SUB 5000
 40 CLS
 41 LET R=0
 42 LET V$='  A A A '
 50 INK 6: PAPER 1: BORDER 1
 51 CLS
 52 LET P=0
 53 LET V=3
 60 CLS
 61 PRINT AT 9,3;'PUNTOS':
 62 PRINT AT 10,4;'0000'
 63 PRINT AT 12,3;'RECORD'
 64 PRINT AT 13,4;'0000'
 70 PRINT  PAPER 7; INK 5;AT 3,13;'433333333333333333
7'
 90 FOR F=4 TO 19
 91 PRINT  INK 5; PAPER 7;AT F,13;'5
5'
 92 NEXT F
 93 PRINT  PAPER 7; INK 5;AT 20,13;'1333333333333333
32'
110 PRINT AT 2,2; INK 5;'EMBOSCADA';TAB 18;V$
120 PLOT 14,150
121 DRAW 75,0
122 DRAW 0,12
123 DRAW -75,0
124 DRAW 0,-12
130 PRINT AT 2,18;'          ';AT 2,18;V$( TO 2*V)
140 LET L1=11
150 LET L2=21
160 PRINT AT 10,4;FN P$(P)
170 PRINT AT 13,4;FN P$(R)
180 PRINT  INK 2; PAPER 7;AT L1,L2;;'A'
181 BEEP .1,0
182 BEEP .1,7
190 LET P=P+10
191 PRINT AT 10,4;FN P$(P)
200 IF R<P THEN  LET R=P: PRINT AT 13,4;FN P$(R)
210 IF CONTA=100 THEN  LET CONTA=0: RESTORE : GO TO
60
220 LET V1=L2+INT (RND*-3)+2
230 LET V2=L1+INT (RND*-3)+2
240 IF V2=L1 AND V1=L2 THEN  GO TO 220
245 IF ATTR (V2,V1)=244 THEN  GO TO 220
250 PRINT  BRIGHT 1; PAPER 6; INK 4; FLASH 1;AT V2,V
1;'B'
251 BEEP .05,7
252 BEEP .05,4
253 BEEP .05,7
254 LET CONTA=CONTA+1
255 GO SUB 600
260 IF SCREEN$ (L1-1,L2)<>' ' THEN  GO TO 280
270 GO TO 330
280 IF SCREEN$ (L1+1,L2)<>' ' THEN  GO TO 300
290 GO TO 330
300 IF SCREEN$ (L1,L2+1)<>' ' THEN  GO TO 320
310 GO TO 330
320 IF SCREEN$ (L1,L2-1)<>' ' THEN  GO TO 3480
330 LET C=1
335 IF INKEY$='' THEN  GO TO 335
336 RESTORE
340 IF L2>15 AND INKEY$='O' THEN  GO TO 400
350 IF L2<28 AND INKEY$='P' THEN  GO TO 420
360 IF L1>5 AND INKEY$='Q' THEN  GO TO 440
370 IF L1<18 AND INKEY$='A' THEN  GO TO 460
380 LET C=C+1
381 IF C=10 THEN  GO TO 220
390 GO TO 340
400 PRINT  PAPER 7;AT L1,L2;' '
401 LET L2=L2-1
402 IF SCREEN$ (L1,L2)<>' ' THEN  GO TO 480
410 GO TO 180
```



Nuestra calabaza posee tres vidas, gracias a lo cual es posible conseguir una buena puntuación.

```
420 PRINT  PAPER 7;AT L1,L2;' '
421 LET L2=L2+1
422 IF SCREEN$ (L1,L2)<>' ' THEN  GO TO 480
430 GO TO 180
440 PRINT  PAPER 7;AT L1,L2;' '
441 LET L1=L1-1
442 IF SCREEN$ (L1,L2)<>' ' THEN  GO TO 480
450 GO TO 180
460 PRINT  PAPER 7;AT L1,L2;' '
461 LET L1=L1+1
462 IF SCREEN$ (L1,L2)<>' ' THEN  GO TO 480
470 GO TO 180
480 FOR F=0 TO 7
481 PRINT  PAPER 7; INK F;AT L1,L2;'A'
482 PAUSE 10
483 NEXT F
484 PRINT AT L1,L2; BRIGHT 1; FLASH 1;'B'
485 LET V=V-1
490 BEEP .3,7
491 BEEP .1,7
492 BEEP .1,7
493 BEEP .1,7
494 BEEP .1,7
495 BEEP .1,9
496 BEEP .2,7
497 BEEP .1,12
498 LET CONTA=0
500 IF V THEN  GO TO 60
510 FOR F=15 TO -15 STEP -1
520 BEEP .01,F
530 NEXT F
540 PRINT AT 11,15; BRIGHT 1; INVERSE 1;'PULSA UNA T
ECLA'
550 RESTORE
560 IF INKEY$='' THEN  GO TO 560
565 LET P=0
570 GO TO 50
600 REM [INV.]  SBR.COMPROBACION [TRUE]
610 IF ATTR (L1-1,L2)=244 AND ATTR (L1+1,L2)=244 AND
 ATTR (L1,L2-1)=244 AND ATTR (L1,L2+1)=244 THEN  GO T
O 630
620 RETURN
630 FOR N=1 TO 10
640 BEEP .05,30
650 BEEP .05,35
660 BEEP .05,40
670 NEXT N
680 PRINT  FLASH 1;AT 21,13;' ESTAS ACORRALADO '
690 PRINT #0;' PULSA UNA TECLA PARA EMPEZAR'
700 IF INKEY$='' THEN  GO TO 700
710 GO TO 50
720 STOP
5000 DATA 102,24,126,219,255
5010 DATA 189,66,60,0,126,126
5020 DATA 126,126,126,126,0
5030 FOR F=0 TO 15
5040 READ A
5050 POKE USR 'A'+F,A
5060 NEXT F
5070 DEF FN P$(A)=('000'+STR$ A)(LEN STR$ A TO )
5080 RETURN
```

En estas páginas finales, hemos pretendido realizar un último esfuerzo de información para nuestros lectores, incluyendo una serie de cuadros, gráficos y tablas, que sin duda les serán de enorme utilidad, a la hora de realizar sus propios programas.
Algunos gráficos han sido incluidos con el fin de que sean fotocopiados y empleados para el diseño de gráficos definidos o pantallas, y otros cuadros o tablas, para que constituyan un sistema de consulta rápida de gran interés.
Esperamos haber satisfecho plenamente y de una forma amena, sus expectativas frente a RUN, Enciclopedia Práctica del Spectrum.

TABLA DE CODIGOS DE ERROR

INFORMES DE LA ROM PRINCIPAL

0	OK	(Correcto)
1	NEXT without FOR	(NEXT sin FOR)
2	Variable not found	(Variable no encontrada)
3	Subscript wrong	(Subíndice erróneo)
4	Out of memory	(Fuera de memoria)
5	Out of screen	(Fuera de pantalla)
6	Number too big	(Número demasiado grande)
7	RETURN without GO SUB	(RETURN sin GO SUB)
8	End of file	(Fin de fichero)
9	STOP statement	(Sentencia STOP)
A	Invalid argument	(Argumento no válido)
B	Integer out of range	(Entero fuera de margen)
C	Nonsense in BASIC	(Sin significado en BASIC)
D	BREAK-CONT repeats	(BREAK con repetición por CONT)
E	Out of DATA	(No quedan datos)
F	Invalid file name	(Nombre de archivo no válido)
G	No room for line	(No hay sitio para la línea)
H	STOP in INPUT	(STOP en entrada por INPUT)
I	FOR without NEXT	(FOR sin NEXT)
J	Invalid I/O device	(Dispositivo de entrada/salida no válido)
K	Invalid colour	(Color no válido)

INFORMES DE LA ROM DEL ZX INTERFACE I

Code error	Error de código
Drive 'write' protected	Unidad protegida contra la escritura
File not found	Fichero no encontrado
Invalid device expresion	Expresión de dispositivo no válida
Invalid drive number	Número de unidad no válido
Invalid name	Nombre no válido
Invalid station number	Número de estación no válido
Invalid stream number	Número de corriente no válido
Merge error	Error de MERGE
Microdrive full	Microdrive completo
Microdrive not present	Microdrive no conectado
Missing baud rate	Falta velocidad de transmisión en baudios
Missing drive number	Falta número de unidad
Missing name	Falta nombre
Missing station number	Falta número de estación
Program finished	Programa terminado
Reading a 'write' file	Intento de escritura en un fichero de 'escritura'
Stream already open	Corriente ya abierta
Verification has failed	Ha fallado la verificación
Writing to a 'read' file	Intento de escritura en un fichero de 'lectura'
Wrong file type	Tipo de fichero equivocado

CONVERSION DECIMAL/HEXA/CARACTER

Dec.	Hex.	Carácter
0	00	no utilizado
1	01	
2	02	
3	03	
4	04	
5	05	
6	06	**PRINT** coma
7	07	**EDIT**
8	08	cursor izquierda
9	09	cursor derecha
10	0A	cursor abajo
11	0B	cursor arriba
12	0C	**DELETE**
13	0D	**ENTER**
14	0E	número
15	0F	no utilizado
16	10	**INK** control
17	11	**PAPER** control
18	12	**FLASH** control
19	13	**BRIGHT** control
20	14	**INVERSE** control
21	15	**OVER** control
22	16	**AT** control
23	17	**TAB** control
24	18	no utilizado
25	19	
26	1A	
27	1B	
28	1C	no utilizado
29	1D	
30	1E	
31	1F	
32	20	espacio
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)

Dec.	Hex.	Carácter
42	2A	*
43	2B	+
44	2C	,
45	2D	—
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	/
93	5D	]
94	5E	↑
95	5F	—
96	60	£
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q

Dec.	Hex.	Carácter
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	©
128	80	□
129	81	◪
130	82	◩
131	83	▨
132	84	◪
133	85	◩
134	86	◪
135	87	◩
136	88	▨
137	89	◪
138	8A	▨
139	8B	◪
140	8C	▨
141	8D	◩
142	8E	◪
143	8F	■
144	90	(a)
145	91	(b)
146	92	(c)
147	93	(d)
148	94	(e)
149	95	(f)
150	96	(g)
151	97	(h)
152	98	(i) gráfico
153	99	(j) definido
154	9A	(k) por el
155	9B	(l) usuario
156	9C	(m)
157	9D	(n)
158	9E	(o)
159	9F	(p)
160	A0	(q)
161	A1	(r)
162	A2	(s)
163	A3	(t)
164	A4	(u)
165	A5	RND
166	A6	INKEY$
167	A7	PI
168	A8	FN
169	A9	POINT
170	AA	SCREEN$
171	AB	ATTR
172	AC	AT
173	AD	TAB
174	AE	VAL$
175	AF	CODE
176	B0	VAL
177	B1	LEN
178	B2	SIN
179	B3	COS
180	B4	TAN
181	B5	ASN
182	B6	ACS
183	B7	ATN
184	B8	LN
185	B9	EXP

Dec.	Hex.	Carácter
186	BA	INT
187	BB	SQR
188	BC	SGN
189	BD	ABS
190	BE	PEEK
191	BF	IN
192	C0	USR
193	C1	STR$
194	C2	CHR$
195	C3	NOT
196	C4	BIN
197	C5	OR
198	C6	AND
199	C7	<=
200	C8	>=
201	C9	<>
202	CA	LINE
203	CB	THEN
204	CC	TO
205	CD	STEP
206	CE	DEF FN
207	CF	CAT
208	D0	FORMAT
209	D1	MOVE
210	D2	ERASE
211	D3	OPEN #
212	D4	CLOSE #
213	D5	MERGE
214	D6	VERIFY
215	D7	BEEP
216	D8	CIRCLE
217	D9	INK
218	DA	PAPER
219	DB	FLASH
220	DC	BRIGHT
221	DD	INVERSE
222	DE	OVER
223	DF	OUT
224	E0	LPRINT
225	E1	LLIST
226	E2	STOP
227	E3	READ
228	E4	DATA
229	E5	RESTORE
230	E6	NEW
231	E7	BORDER
232	E8	CONTINUE
233	E9	DIM
234	EA	REM
235	EB	FOR
236	EC	GO TO
237	ED	GO SUB
238	EE	INPUT
239	EF	LOAD
240	F0	LIST
241	F1	LET
242	F2	PAUSE
243	F3	NEXT
244	F4	POKE
245	F5	PRINT
246	F6	PLOT
247	F7	RUN
248	F8	SAVE
249	F9	RANDOMIZE
250	FA	IF
251	FB	CLS
252	FC	DRAW
253	FD	CLEAR
254	FE	RETURN
255	FF	COPY

VARIABLES DEL SISTEMA

Bytes	Dir. Inicio	Nombre	Utilización
ROM PRINCIPAL			
8	23552	KSTATE	Lectura del teclado.
1	23560	LAST K	Ultima tecla pulsada.
1	23561	REPDEL	Tiempo de pulsación de una tecla antes de su repetición automática (en cincuentavos de segundo). Valor por defecto: 35.
1	23562	REPPER	Retardo entre repeticiones de una tecla.
2	23563	DEFADD	Dirección del argumento de una función definida por el usuario.
1	23565	K DATA	Segundo byte de control de color introducido por el teclado.
2	23566	TVDATA	Controles del color, **AT** y **TAB** a la pantalla.
38	23568	STRMS	Direcciones de los canales de comunicación de entrada y salida.
2	23606	CHARS	256 bytes antes del puntero del generador de caracteres. Valor por defecto: 15360.
1	23608	RASP	Duración del tono de alarma.
1	23609	PIP	Duración del 'click' del teclado.
1	23610	ERR NR	Ultimo código de error menos 1.
1	23611	FLAGS	Varias banderas.
1	23612	TVFLAG	Banderas asociadas con el televisor.
2	23613	ERR SP	Dirección del *stack* en la cual se encuentra la dirección de retorno por error.
2	23615	LISTSP	Dirección de la posición de retorno tras un listado automático.
1	23617	MODE	Modo del cursor (K, L, C, E o G).
2	23618	NEWPPC	Próxima línea a ejecutar.
1	23620	NSPPC	Próxima sentencia dentro de la línea a ejecutar.
2	23621	PPC	Número de línea de la sentencia en ejecución.
1	23623	SUBPPC	Número de sentencia dentro de la línea en ejecución.
1	23624	BORDCR	Color del **BORDER***8 y atributos empleados por defecto para la zona inferior de la pantalla.
2	23625	E PPC	Número de línea en que se encuentra el cursor de programa.
2	23627	VARS	Dirección de inicio del área de variables.
2	23629	DEST	Dirección de variable en asignación.
2	23631	CHANS	Dirección del canal de datos.
2	23633	CURCHL	Dirección del canal que se utiliza para entrada/salida de información.
2	23635	PROG	Dirección de inicio del área de texto BASIC.
2	23637	NXTLIN	Dirección de la siguiente línea del programa.
2	23639	DATADD	Dirección de la terminación del último elemento de datos.
2	23641	E LINE	Dirección del área de introducción BASIC.
2	23643	K CUR	Dirección del cursor.
2	23645	CH ADD	Dirección del próximo carácter a interpretar.
2	23647	X PTR	Dirección del carácter que sigue al de sintaxis errónea (interrogación en **FLASH**).
2	23649	WK SPC	Dirección del área de trabajo.
2	23651	STKBOT	Dirección de inicio de la pila de cálculo.
2	23653	STKEND	Dirección de inicio del área de reserva.
1	23655	BREG	Registro B del calculador.
2	23656	MEM	Dirección de la zona empleada como memoria del calculador.
1	23658	FLAGS2	Indicador de mayúsculas (8) o minúsculas.
1	23659	DF SZ	Número de líneas de la zona inferior de la pantalla.
2	23660	S TOP	Número de línea superior en la pantalla de listado automático.
2	23662	OLDPPC	Número de línea en que continúa la sentencia **CONT.**
1	23664	OSPPC	Número de sentencia dentro de la línea en que continúa la sentencia **CONT.**
1	23665	FLAGX	Diversas banderas.
2	23666	STRLEN	Espacio ocupado por la cadena en asignación.
2	23668	T ADDR	Dirección del siguiente elemento en la tabla de sintaxis.
2	23670	SEED	Semilla para **RND.**
3	23672	FRAMES	Contador de cuadros de TV (incrementado cada 20 ms).
2	23675	UDG	Dirección del generador de caracteres de usuario (UDG).
1	23677	COORDX	Coordenada X del último punto impreso en alta resolución.
1	23678	COORDY	Coordenada Y del último punto impreso en alta resolución.
1	23679	P POSN	Columna de posición de la impresora.
1	23680	PR CC	Byte menos significativo de la siguiente posición a imprimir en el *buffer* de la impresora.
1	23681	FREE	Byte libre para el usuario no afectado por **NEW.**
2	23682	ECHO E	Número de las 33 columnas y de las 24 líneas (en la zona inferior) del final de la memoria temporal de entrada.
2	23684	DF CC	Dirección de la posición de **PRINT** en el fichero de presentación visual.
2	23686	DFCCL	Igual que DFCC, pero para la zona inferior de la pantalla.
1	23688	S POSN	Número de las 33 columnas de la posición de **PRINT.**

Bytes	Dir. Inicio	Nombre	Utilización
1	23689	SPOSNL	Número de las 24 líneas de la posición de **PRINT.**
1	23692	SCR CT	Contador de líneas para *scrolling.*
1	23693	ATTR P	Atributos por defecto fijados por comandos.
1	23694	MASK P	Máscara para creación de colores «transparentes» (código 8).
1	23695	ATTR P	Atributos de uso temporal, fijados mediante sentencias **PRINT.**
1	23696	MASK T	Máscara para la «transparencia» en colores temporales.
1	23697	P FLAG	Banderas.
30	23698	MEMBOT	Memoria del calculador.
2	23728	D FREE	Pareja de bytes libres para el usuario, no afectados por **NEW.**
2	23730	RAMTOP	Tope máximo de la zona de memoria empleable por el BASIC.
2	23732	P RAMT	Tope máximo de la RAM física.
ROM DEL INTERFACE 1			
1	23734	FLAG 3	Banderas.
2	23735	VECTOR	Dirección de ampliación del intérprete BASIC.
10	23737	SBRT	Subrutina de paginación ROM.
2	23747	BAUD	Velocidad de transmisión en baudios para el interface RS-232 (BAUD=(3500000/(26*vel.trans.baud.))–2.
1	23749	NTSTAT	Número de estación de red propia.
1	23750	IOBORD	Color empleado para **BORDER** durante las operaciones de entrada/salida.
2	23751	SER FL	Zona de trabajo del RS-232.
2	23753	SECTOR	Zona de trabajo del Microdrive.
2	23755	CHADD	Almacenamiento temporal para CH ADD.
1	23757	NTRESP	Código de respuesta de red.
1	23758	NTDEST	*Buffer* de red.
1	23759	NTSRCE	Número de estación fuente de red.
2	23760	NTNUMB	Número de bloque de red.
1	23762	NTTYPE	Código tipo de encabezamiento.
1	23763	NTLEN	Longitud de bloque de datos.
1	23764	NTDCS	*Checksum* de bloque de datos.
1	23765	NTHCS	*Checksum* de bloque de encabezamientos.
1	23766	D STR1	Comienzo de especificador de fichero de 8 bytes.
1	23767	C STR1	Número de unidad de control.
1	23768	S STR1	Número de corriente.
1	23769	L STR1	Tipo de dispositivo (m, n, t o b).
2	23770	N STR1	Longitud de nombre de fichero.
2	23772	B STR1	Comienzo de nombre de fichero.
8	23774	D STR2	Segundo especificador de fichero utilizado por **MOVE** y **LOAD.**
1	23782	HD 00	Comienzo de zona de trabajo para **SAVE, LOAD, VERIFY** y **MERGE.**
2	23783	HD 0B	Longitud de datos.
2	23785	HD 0D	Comienzo de datos.
2	23787	HD 0F	Longitud de programa.
2	23789	HD 11	Número de línea.
1	23791	COPIES	Número de copias a efectuar por **SAVE.**

MNEMONICOS Z-80

Dec	Hex.	Bin.	Directo	Por CB	Por ED	Por DD	Por FD	Por DDCB	Por FDCB
0	00	0	nop	rlc b					
1	01	1	ld bc,NN	rlc c					
2	02	10	ld (bc),a	rlc d					
3	03	11	inc bc	rlc e					
4	04	100	inc b	rlc h					
5	05	101	dec b	rlc l					
6	06	110	ld b,N	rlc (hl)				rlc (ix+N)	rlc (iy+N)
7	07	111	rlca	rlc a					
8	08	1000	ex af,af	rrc b					
9	09	1001	add hl,bc	rrc c		add ix,bc	add iy,bc		

Dec	Hex.	Bin.	Directo	Por CB	Por ED	Por DD	Por FD	Por DDCB	Por FDCB
10	0A	1010	ld a,(bc)	rrc d					
11	0B	1011	dec bc	rrc e					
12	0C	1100	Inc c	rrc h					
13	0D	1101	dec c	rrc l					
14	0E	1110	ld c,N	rrc (hl)				rrc (ix+N)	rrc (iy+N)
15	0F	1111	rrca	rrca					
16	10	10000	djnz DIS	rl b					
17	11	10001	ld de,NN	rl c					
18	12	10010	ld (de),a	rl d					
19	13	10011	inc de	rl e					
20	14	10100	inc d	rl h					
21	15	10101	dec d	rl l					
22	16	10110	ld d,N	rl (hl)				rl (ix+N)	rl (iy+N)
23	17	10111	rla	rl a					
24	18	11000	jr DIS	rr b					
25	19	11001	add hl,de	rr c		add ix, de	add iy,de		
26	1A	11010	ld a,(de)	rr d					
27	1B	11011	dec de	rr e					
28	1C	11100	inc e	rr h					
29	1D	11101	dec e	rr l					
30	1E	11110	ld e,N	rr (hl)				rr (ix+N)	rr (iy+N)
31	1F	11111	rra	rr a					
32	20	100000	jr nz,DIS	sla b					
33	21	100001	ld hl,NN	sla c					
34	22	100010	ld (NN),hl	sla d		ld (NN),ix	ld (NN),iy		
35	23	100011	inc hl	sla e		inc ix	inc iy		
36	24	100100	inc h	sla h					
37	25	100101	dec h	sla l					
38	26	100110	ld h,N	sla (hl)				sla (ix+N)	sla (iy+N)
39	27	100111	daa	sla a					
40	28	101000	jr z,DIS	sra b					
41	29	101001	add hl,hl	sra c		add ix,ix	add iy,iy		
42	2A	101010	ld hl,(NN)	sra d		ld ix,(NN)	ld iy,(NN)		
43	2B	101011	dec hl	sra e		dec ix	dec iy		
44	2C	101100	inc l	sra h					
45	2D	101101	dec l	sra l					
46	2E	101110	ld l,N	sra (hl)				sra (ix+N)	sra (iy+N)
47	2F	101111	cpl	sra a					
48	30	110000	jr nc,DIS						
49	31	110001	ld sp,NN						
50	32	110010	ld (NN),a						
51	33	110011	inc sp						
52	34	110100	inc (hl)			ine (ix+N)	ine (iy+N)		
53	35	110101	dec (hl)			dec (ix+N)	dec (iy+N)		
54	36	110110	ld (hl),N			ld (ix+N),N	ld (iy+N),N		
55	37	110111	scf						
56	38	111000	jr c,DIS	srl b					
57	39	111001	add hl,sp	srl c		add ix,sp	add ix,sp		
58	3A	111010	ld a,(NN)	srl d					
59	3B	111011	dec sp	srl e					
60	3C	111100	inc a	srl h					
61	3D	111101	dec a	srl l					
62	3E	111100	ld a,N	srl (hl)				srl (ix+N)	srl (iy+N)
63	3F	111111	ccf	srl a					
64	40	1000000	ld b,b	bit 0,b	in b,(c)				
65	41	1000001	ld b,c	bit 0,c	out (c),b				
66	42	1000010	ld b,d	bit 0,d	sbc hl,bc				
67	43	1000011	ld b,e	bit 0,e	ld (NN),bc				
68	44	1000100	ld b,h	bit 0,h	neg				
69	45	1000101	ld b,l	bit 0,l	retn				
70	46	1000110	ld b,(hl)	bit 0,(hl)	im 0	ld b, (ix+N)	ld b,(iy+N)	bit 0, (ix+N)	bit 0,(iy+N)
71	47	1000111	ld b,a	bit 0,a	ld i,a				
72	48	1001000	ld c,b	bit 1,b	in c,(c)				
73	49	1001001	ld c,c	bit 1,c	out (c),c				
74	4A	1001010	ld c,d	bit 1,d	adc hl,bc				
75	4B	1001011	ld c,e	bit 1,e	ld bc,(NN)				
76	4C	1001100	ld c,h	bit 1,h					
77	4D	1001101	ld c,l	bit 1,l	reti				

Dec	Hex.	Bin.	Directo	Por CB	Por ED	Por DD	Por FD	Por DDCB	Por FDCB
78	4E	1001110	ld c,(hl)	bit 1,(hl)		ld c,(ix+N)	ld c,(iy+N)	bit 1,(ix+N)	bit 1,(iy+N)
79	4F	1001111	ld c,a	bit 1,a	ld r,a				
80	50	1010000	ld d,b	bit 2,b	in d,(c)				
81	51	1010001	ld d,c	bit 2,c	out (c),d				
82	52	1010010	ld d,d	bit 2,d	sbc hl,de				
83	53	1010011	ld d,e	bit 2,e	ld (NN),de				
84	54	1010100	ld d,h	bit 2,h					
85	55	1010101	ld d,l	bit 2,l					
86	56	1010110	ld d,(hl)	bit 2,(hl)	im 1	ld d,(ix+N)	ld d,(iy+N)	bit 2, (ix+N)	bit 2,(iy+N)
87	57	1010111	ld d,a	bit 2,a	ld a,i				
88	58	1011000	ld e,b	bit 3,b	in e,(c)				
89	59	1011001	ld e,c	bit 3,c	out (c),e				
90	5A	1011010	ld e,d	bit 3,d	adc hl,de				
91	5B	1011011	ld e,e	bit 3,e	ld de,(NN)				
92	5C	1011100	ld e,h	bit 3,h					
93	5D	1011101	ld e,l	bit 3,l					
94	5E	1011110	ld e,(hl)	bit 3,(hl)	im 2	ld e,(ix+N)	ld e,(iy+N)	bit 3,(ix+N)	bit 3,(iy+N)
95	5F	1011111	ld e,a	bit 3,a	ld a,r				
96	60	1100000	ld h,b	bit 4,b	in h,(c)				
97	61	1100001	ld hc,	bit 4,c	out (c),h				
98	62	1100010	ld h,d	bit 4,d	sbc hl,hl				
99	63	1100011	ld h,e	bit 4,e	ld (NN),hl				
100	64	1100100	ld h,h	bit 4,h					
101	65	1100101	ld h,l	bit 4,l					
102	66	1100110	ld h,(hl)	bit 4,(hl)		ld h,(ix+N)	ld h,(iy+N)	bit y,(ix+N)	bit 4,(iy+N)
103	67	1100111	ld h,a	bit 4,a	rrd				
104	68	1101000	ld l,b	bit 5,b	in l,(c)				
105	69	1101001	ld l,c	bit 5,c	out (c),l				
106	6A	1101010	ld l,d	bit 5,d	adc hl,hl				
107	6B	1101011	ld l,e	bit 5,e	ld hl,(NN)				
108	6C	1101100	ld l,h	bit 5,h					
109	6D	1101101	ld l,l	bit 5,l					
110	6E	1101110	ld l,(hl)	bit 5,(hl)		la l,(ix+N)	ld l,(iy+N)	bit 5,(ix+N)	bit 5,(iy+N)
111	6F	1101111	ld l,a	bit 5,a	rld				
112	70	1110000	ld (hl),b	bit 6,b	in f,(c)	ld (ix+N)	ld (iy+N),b		
113	71	1110001	ld (hl),c	bit 6,c		ld (ix+N),c	ld (iy+N),c		
114	72	1110010	ld (hl),d	bit 6,d	sbc hl,sp	id (ix+N),d	ld (iy+N),d		
115	73	1110011	ld (hl),e	bit 6,e	ld (NN),sp	ld (ix+N),e	ld (iy+N),e		
116	74	1110100	ld (hl),h	bit 6,h		ld (ix+N),h	ld (iy+N),h		
117	75	1110101	ld (hl),l	bit 6,l		ld (ix+N),l	ld (iy+N),l		
118	76	1110110	halt	bit 6,(hl)				bit 6,(ix+N)	bit 6,(iy+N)
119	77	1110111	ld (hl),a	bit 6,a		ld (ix+N), a	ld (iy+N),a		
120	78	1111000	ld a,b	bit 7,b	in a,(c)				
121	79	1111001	ld a,c	bit 7,c	out (c),a				
122	7A	1111010	ld a,d	bit 7,d	adc hl,sp				
123	7B	1111011	ld a,e	bit 7,e	ld sp,(NN)				
124	7C	1111100	ld a,h	bit 7,h					
125	7D	1111101	ld a,l	bit 7,l					
126	7E	1111110	ld a,(hl)	bit 7,(hl)		ld a,(ix+N)	ld a,(iy+N)	bit 7,(ix+N)	bit 7,(iy+N)
127	7F	1111111	ld a,a	bit 7,a					
128	80	1000000	add a,b	res 0,b					
129	81	1000001	add a,c	res 0,c					
130	82	10000010	add a,d	res 0,d					
131	83	10000011	add a,e	res 0,e					
132	84	10000100	add a,h	res 0,h					
133	85	10000101	add a,l	res 0,l					
134	86	10000110	add a,(hl)	res 0,(hl)		add a,(ix+N)	add a,(iy+N)	res o,(ix+N)	res o,(iy+N)
135	87	10000111	add a,a	res 0,a					
136	88	10001000	adc a,b	res 1,b					
137	89	10001001	adc a,c	res 1,c					
138	8A	10001010	adc a,d	res 1,d					
139	8B	10001011	adc a,e	res 1,e					
140	8C	10001100	adc a,h	res 1,h					
141	8D	10001101	adc a,l	res 1,l					
142	8E	10001110	adc a,(hl)	res 1,(hl)		adc a,(ix+N)	adc a,(iy+N)	res 1,(ix+N)	res 1,(iy+N)
143	8F	10001111	adc a,a	res 1,a					
144	90	10010000	sub b	res 2,b					
145	91	10010001	sub c	res 2,c					
146	92	10010010	sub d	res 2,d					

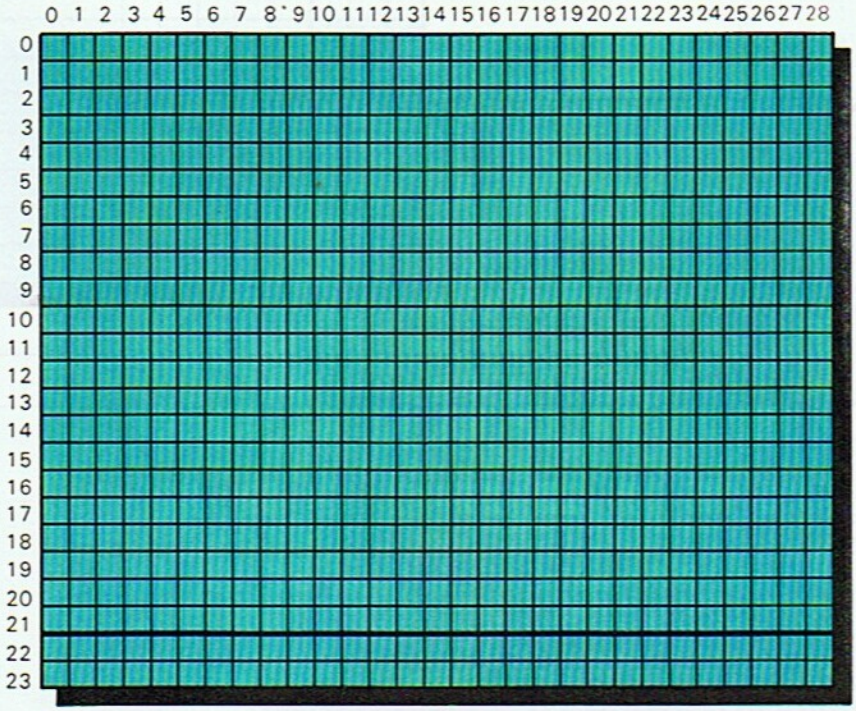
Dec	Hex.	Bin.	Directo	Por CB	Por ED	Por DD	Por FD	Por DDCB	Por FDCB
147	93	10010011	sub e	res 2,e					
148	94	10010100	sub h	res 2,h					
149	95	10010101	sub l	res 2,l					
150	96	10010110	sub (hl)	res 2,(hl)		sub (ix+N)	sub (iy+N)	res 2,(ix+N)	res 2,(iy+N)
151	97	10010111	sub a	res 2,a					
152	98	10011000	sbc a,b	res 3,b					
153	99	10011001	sbc a,c	res 3,c					
154	9A	10011010	sbc a,d	res 3,d					
155	9B	10011011	sbc a,e	res 3,e					
156	9C	10011100	sbc a,h	res 3,h					
157	9D	10011101	sbc a,l	res 3,l					
158	9E	10011110	sbc a,(hl)	res 3,(hl)		sbc a,(ix+N)	sbc a,(iy+N)	res 3,(ix+N)	res 3,(iy+N)
159	9F	10011111	sbc a,a	res 3,a					
160	A0	10100000	and b	res 4,b	ldi				
161	A1	10100001	and c	res 4,c	cpi				
162	A2	10100010	and d	res 4,d	ini				
163	A3	10100011	and e	res 4,e	outi				
164	A4	10100100	and h	res 4,h					
165	A5	10100101	and l	res 4,l					
166	A6	10100110	and (hl)	res 4,(hl)		and (ix+N)	and (iy+N)	res 4,(ix+N)	res 4,(iy+N)
167	A7	10100111	and a	res 4,a					
168	A8	10101000	xor b	res 5,b	ldd				
169	A9	10101001	xor c	res 5,c	cpd				
170	AA	10101010	xor d	res 5,d	ind				
171	AB	10101011	xor e	res 5,e	outd				
172	AC	10101100	xor h	res 5,h					
173	AD	10101101	xor l	res 5,l					
174	AE	10101110	xor (hl)	res 5,(hl)		xor (ix+N)	xor (iy+N)	res 5,(ix+N)	res 5,(iy+N)
175	AF	10101111	xor a	res 5,a					
176	B0	10110000	or b	res 6,b	ldir				
177	B1	10110001	or c	res 6,c	cpir				
178	B2	10110010	or d	res 6,d	inir				
179	B3	10110011	or e	res 6,e	otir				
180	B4	10110100	or h	res 6,h					
181	B5	10110101	or l	res 6,l					
182	B6	10110110	or (hl)	res 6,(hl)		or (ix+N)	or (iy+N)	res 6,(ix+N)	res 6,(iy+N)
183	B7	10110111	or a	res 6,a					
184	B8	10111000	cp b	res 7,b	lddr				
185	B9	10111001	cp c	res 7,c	cpdr				
186	BA	10111010	cp d	res 7,d	indr				
187	BB	10111011	cp e	res 7,e	otdr				
188	BC	10111100	cp h	res 7,h					
189	BD	10111101	cp l	res 7,l					
190	BE	10111110	cp (hl)	res 7,(hl)		cp (ix+N))	cp (iy+N)	res 7,(ix+N)	res 7,(iy+N)
191	BF	10111111	cp a	res 7,a					
192	C0	11000000	ret nz	set 0,b					
193	C1	11000001	pop bc	set 0,c					
194	C2	11000010	jp nz,NN	set 0,d					
195	C3	11000011	jp NN	set 0,e					
196	C4	11000100	call nz,NN	set 0,h					
197	C5	11000101	push bc	set 0,l				set o,(ix+N)	set o,(iy+N)
198	C6	11000110	add a,N	set 0,(hl)					
199	C7	11000111	rst 0	set 0,a					
200	C8	11001000	ret z	set 1,b					
201	C9	11001001	ret	set 1,c					
202	CA	11001010	jp z,NN	set 1,d					
203	CB	11001011		set 1,e					
204	CC	11001100	call z,NN	set 1,h					
205	CD	11001101	call NN	set 1,l				set 1,(ix+N)	set 1,(iy+N)
206	CE	11001110	adc a,N	set 1,(hl)					
207	CF	11001111	rst 8	set 1,a					
208	D0	11010000	ret nc	set 2,b					
209	D1	11010001	pop de	set 2,c					
210	D2	11010010	jp nc,NN	set 2,d					
211	D3	11010011	out (N),a	set 2,e					
212	D4	11010100	call nc,NN	set 2,h					
213	D5	11010101	push de	set 2,l					
214	D6	11010110	sub N	set 2,(hl)				set 2,(ix+N)	set 2,(iy+N)
215	D7	11010111	rst 16	set 2,a					

Dec	Hex.	Bin.	Directo	Por CB	Por ED	Por DD	Por FD	Por DDCB	Por FDCB
216	D8	11011000	ret c	set 3,b					
217	D9	11011001	exx	set 3,c					
218	DA	11011010	jp c.NN	set 3,d					
219	DB	11011011	in a,(N)	set 3,e					
220	DC	11011100	call c,NN	set 3,h					
221	DD	11011101		set 3,l					
222	DE	11011110	sbc a,N	set 3,(hl)				set 3,(ix+N)	set 3,(iy+N)
223	DF	11011111	rst 24	set 3,a					
224	E0	11100000	ret po	set 4,b					
225	E1	11100001	pop hl	set 4,c		pop ix	pop iy		
226	E2	11100010	jp po,NN	set 4,d					
227	E3	11100011	ex (sp),hl	set 4,e		ex (sp),ix	ex (sp),iy		
228	E4	11100100	call po,NN	set 4,h					
229	E5	11100101	push hl	set 4,l		pus ix	push iy		
230	E6	11100110	and N	set 4,(hl)				set 4,(ix+N)	set 4,(iy+N)
231	E7	11100111	rst 32	set 4,a					
232	E8	11101000	ret pe	set 5,b					
233	E9	11101001	jp (hl)	set 5,c		jp (ix)	jp (iy)		
234	EA	11101010	jp pe,NN	set 5,d					
235	EB	11101011	ex de,hl	set 5,e					
236	EC	11101100	call pe,NN	set 5,h					
237	ED	11101101		set 5,l					
238	EE	11101110	xor N	set 5,(hl)				set 5,(ix+N)	set 5,(iy+N)
239	EF	11101111	rst 40	set 5,a					
240	F0	11110000	ret p	set 6,b					
241	F1	11110001	pop af	set 6,c					
242	F2	11110010	jp p,NN	set 6,d					
243	F3	11110011	di	set 6,e					
244	F4	11110100	call p,NN	set 6,h					
245	F5	11110101	push af	set 6,l					
246	F6	11110110	or N	set 6,(hl)				set 6,(ix+N)	set 6,(iy+N)
247	F7	11110111	rst 48	set 6,a					
248	F8	11111000	ret m	set 7,b					
249	F9	11111001	ld sp,hl	set 7,c		ld sp, ix	ld sp, iy		
250	FA	11111010	jp m,NN	set 7,d					
251	FB	11111011	ei	set 7,e					
252	FC	11111100	call m,NN	set 7,h					
253	FD	11111101		set 7,l					
254	FE	11111110	cp N	ser 7,(hl)				set 7,(ix+N)	set 7,(iy+N)
255	FF	11111111	rst 56	set 7,a					

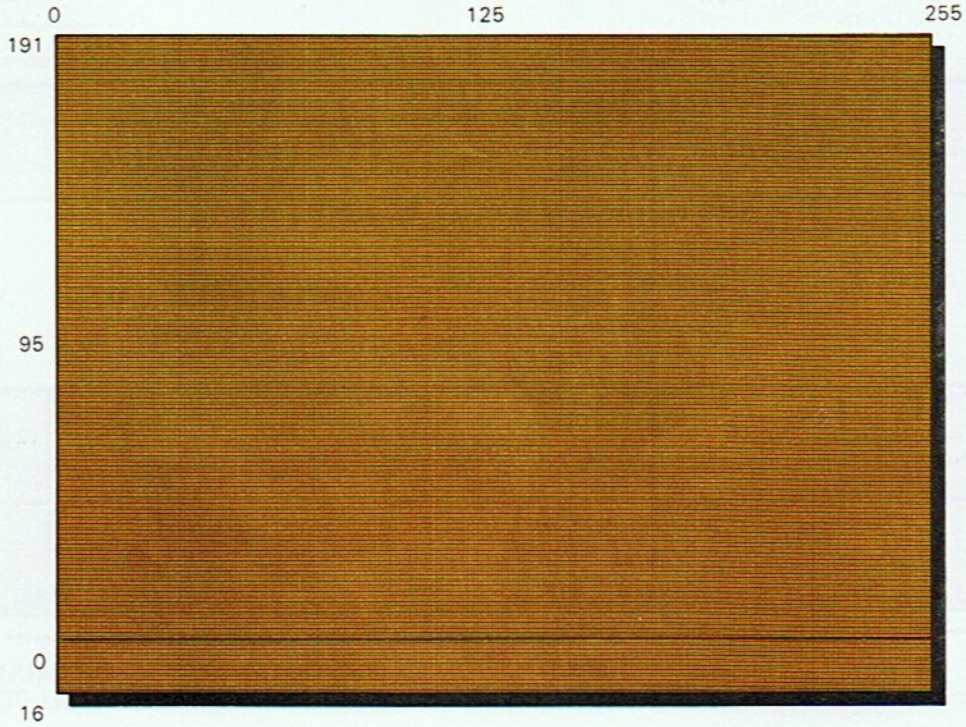
TABLA DE PRIORIDADES

Operación	Prioridad
Asignación de subíndices y fragmentación	12
Todas las funciones salvo **NOT** y operación unaria 'menos'	11
Potenciación	10
Operación unaria 'menos' (es decir, 'menos' sólo usado para negar algo)	9
*, /	8
+, – ('menos' usado para restar un número de otro)	6
=, >, <, <=, >=, <>	5
NOT	4
AND	3
OR	2

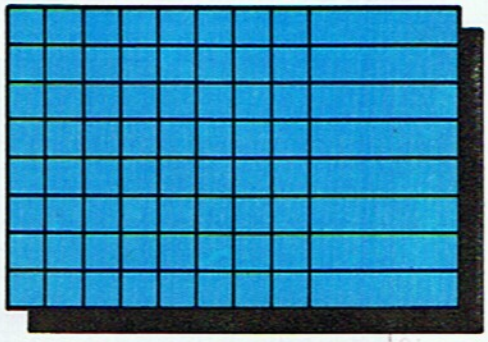
BAJA RESOLUCION



ALTA RESOLUCION

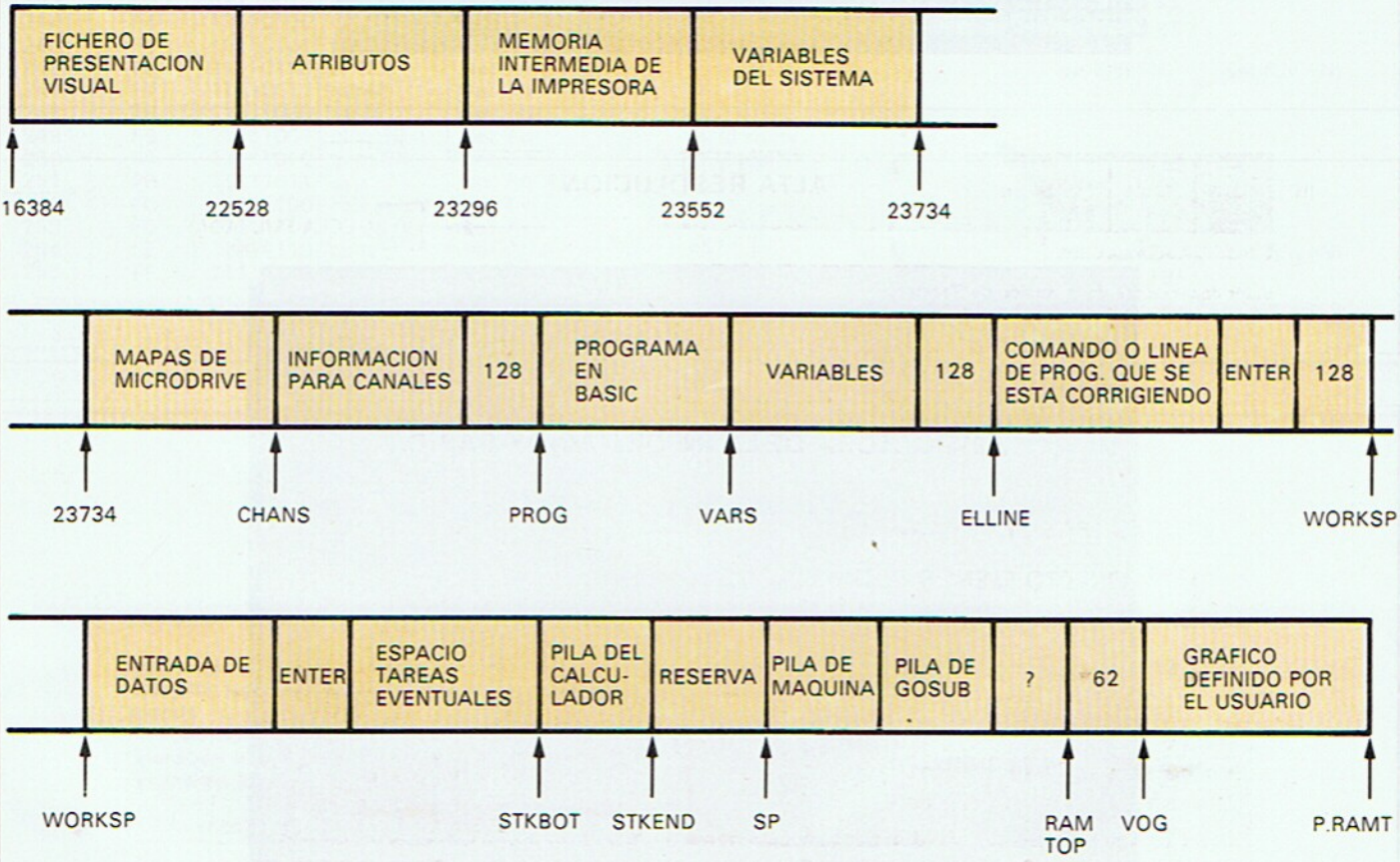


HOJA DE DISEÑO DE CARACTERES

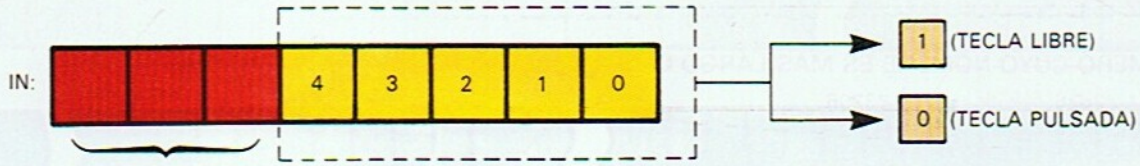


TITULO:
GRAFICO:
PROGRAMA:

MAPA DE MEMORIA



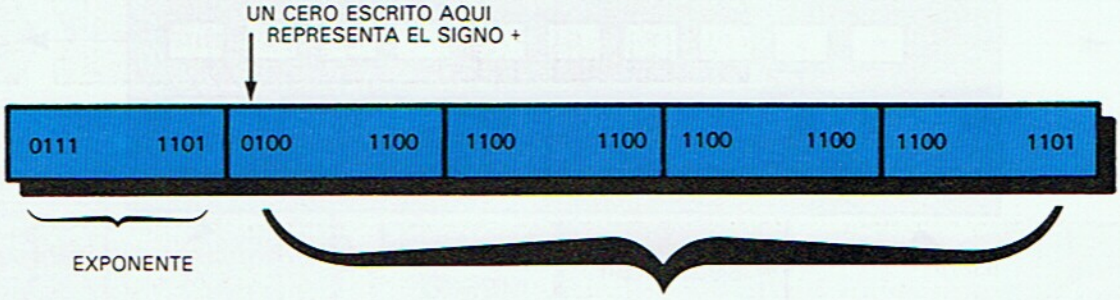
LECTURA DEL TECLADO CON IN



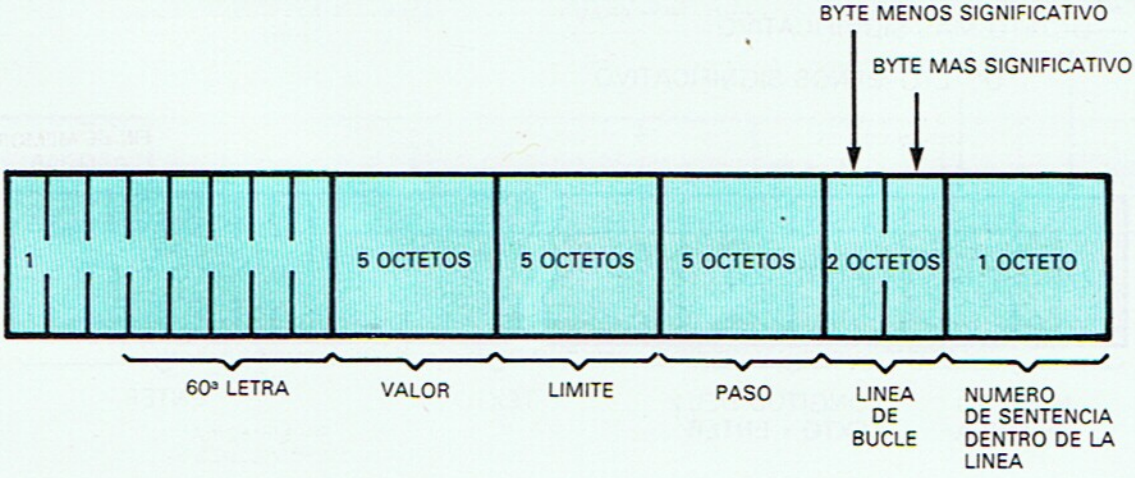
Valor dependiente de la versión de Spectrum y no del resultado de la función IN.

ESTRUCTURA DE LA INFORMACION BASICA

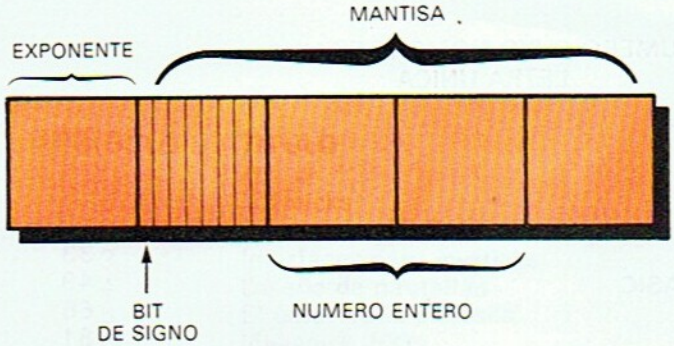
ESTRUCTURA DE LA INFORMACION BASICA

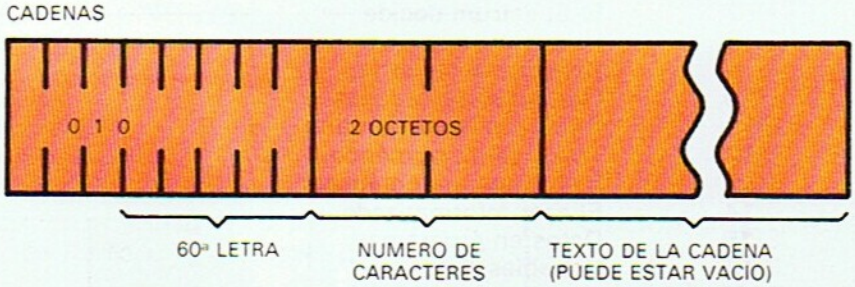


NUMERO CUYO NOMBRE ES MAS LARGO QUE UNA LETRA:

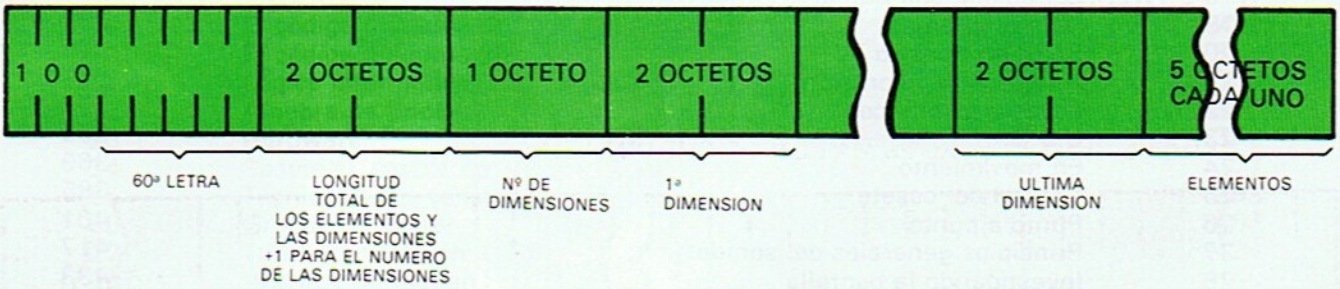


ESTRUCTURA DE LA INFORMACION BASICA

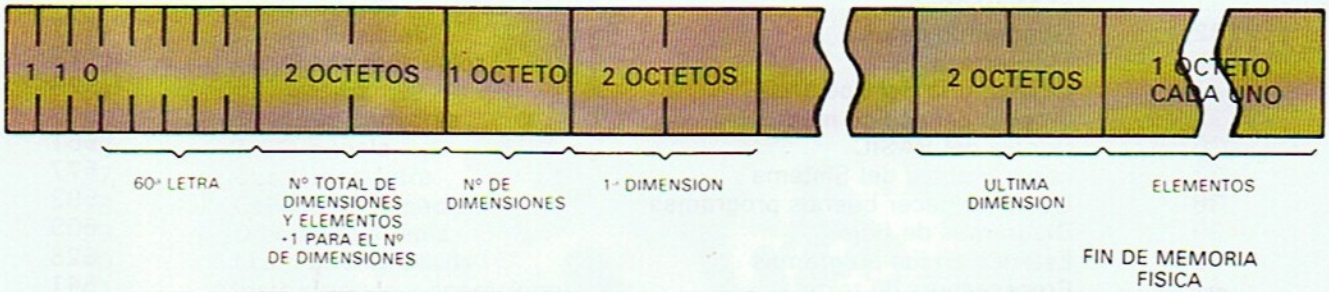


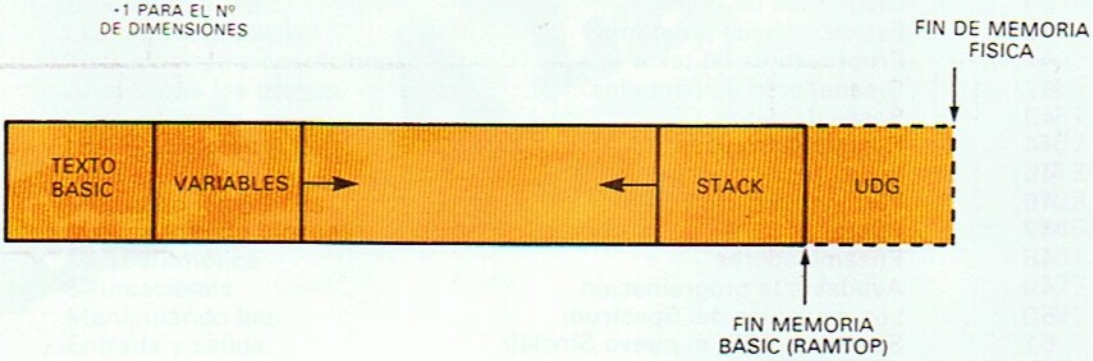


MATRIZ DE NUMEROS



MATRIZ DE CARACTERES:





INDICE BASIC

FASCICULO	TITULO	PAGINA
1	El Spectrum	1
2	Dominando el teclado	17
3	Una supercalculadora	33
4	Principios de programación en BASIC	49
5	Manejo del casete	65
6	Un poco de práctica	81
7	El Spectrum decide	97
8	Una y otra vez	113
9	Rompiendo cadenas	129
10	Evaluador de cadenas	145
11	Situación en la pantalla	161
12	Tratamiento numérico	177
13	Funciones matemáticas	193
14	El azar informatizado	209
15	Datos en programa	225
16	Variables suscritas	241
17	Prácticas con matrices	257
18	Letras que son números	273
19	Mini programas	289
20	Color en nuestro TV	305
21	Atributos de impresión	321
22	Caracteres gráficos	337
23	U.D.G.	353
24	En movimiento	369
25	Control del casete	385
26	Punto a punto	401
27	Principios generales del sonido	417
28	Investigando la pantalla	433
29	Más de un jugador	449
30	La impresora	465
31	El Microdrive	481
32	Guardando datos	497
33	El Interface 1	513
34	La memoria del Spectrum	529
35	Empleo del código máquina	545
36	Dentro del BASIC	561
37	Las Variables del Sistema	577
38	De como hacer buenos programas	593
39	Diagramas de flujo	609
40	Estética en los programas	625
41	Procesadores de texto	641
42	Diseñadores de pantallas	657
43	Bases de datos	673
44	Hojas electrónicas	689
45	Logo	705
46	Forth	721
47	Pascal	737
48	Ensambladores	753
49	Ayudas a la programación	769
50	Los mensajes del Spectrum	785
51	Spectrum 128: el nuevo Sinclair	801

INDICE TU SPECTRUM

FASCICULO	TITULO	PAGINA
1	Joysticks	9
2	Protección del Software	24
3	Interfaces para joysticks	41
4	Diseño de pantallas	57
5	El color de la pantalla	73
6	Invesdisk 200	88
7	El T.O.S.	104
8	Los sorts	120
9	Impresoras	136
10	Sistemas de numeración	153
11	El interior del Spectrum	168
12	El mapa del Spectrum	185
13	Dinamismo de pantallas	201
14	A toda pantalla	216
15	Gráficos animados	233
16	¡Se rueda!	249
17	Las impresoras del Spectrum	265
18	Manejo de series	281
19	El código máquina (I)	297
20	El código máquina (II)	313
21	Lógica informática	330
22	Algebra de Boole	345
23	Hardware	362
24	Casete sí, casete no	377
25	Técnica de los casetes	393
26	Grabación digital	409
27	Turbo transmisión	424
28	Una buena imagen	440
29	Videoterminales	456
30	Entre dos eras	472
31	Procesadores de ideas	489
32	Discos flexibles	505
33	La jugada más veloz	521
34	En alta voz	537
35	Tu micro conversa	553
36	Tecla a tecla	569
37	Con buena letra	585
38	Calidad de impacto	601
39	Dibujando a lápiz	616
40	Los primeros pasos	633
41	Instrucciones y operaciones	646
42	Analizando los grupos	665
43	Direccionando	681
44	Los indicadores	697
45	De 8 en 8 bits	713
46	Moviendo los 16 bits	729
47	Movimiento de bloques	745
48	Microaritmética	761
49	Bifurcaciones	777
50	Manipulando bits	793
51	Entrada y salida	809

INDICE PROGRAMA

FASCICULO	TITULO	PAGINA
1	Hundido	13
2	Funciones 3D	30
3	Cangurín	47
4	Saturno jet	62
5	Adivinador	78
6	Tele-Sketch	94
7	Aventura interminable	109
8	Modulo lunar	126
9	Progresiones aritméticas	141
10	La bomba	157
11	Master cubos	173
12	Antiaéreo	191
13	¡Bingo!	207
14	Arañas	221
15	Acorazado	238
16	Simón	253
17	El cubo mágico	269
18	Index	285
19	Caracteres gigantes	301
20	Black Jack	317
21	Biorritmos	334
22	Puzzle	350
23	El trueno azul	367
24	El profesor	382
25	Quebradero	398
26	Motos de luz	413
27	Turbo	430
28	El elefante	444
29	Motora	461
30	Damas chinas	478
31	La presa	494
32	Las cuatro en raya	510
33	Longevidad	525
34	Gusy	543
35	Hunter	558
36	Solitario	574
37	Evasión	590
38	Posters	607
39	Magnet	622
40	Trigonometría	638
41	Imprenta	654
42	Salto de caballo	670
43	Nim	686
44	Tanques	702
45	Morse	718
46	Tragaperras	734
47	Radar	750
48	Paleta	766
49	Calimocho	782
50	Tira pared	798
51	Emboscada	814